Network Working Group
Internet-Draft
Updates: 5357, 7750 (if approved)
Intended status: Standards Track
Expires: August 27, 2016

S. Baillargeon
G. Mirsky
Ericsson

February 24, 2016

Control and Monitoring Differentiated Service Code Point in Two-Way Active Measurement Protocol (TWAMP) draft-bailmir-ippm-twamp-dscp-ctrl-mon-00

Abstract

This document describes an optional extension for Two-Way Active Measurement Protocol (TWAMP) allowing control and monitoring of the Differentiated Service Code Point (DSCP) field in forward and reverse directions within single test session with the TWAMP-Test protocol. This document, if accepted, will be an update to the TWAMP core protocol specified in RFC 5357 and DSCP Monitoring mode defined in RFC 7750 .

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, 2016.

Table of Contents

1. Introduction

The One-Way Active Measurement Protocol (OWAMP) [RFC4656] defines the Type-P Descriptor field and negotiation of its value in OWAMP-Control protocol. The Two-Way Active Measurement Protocol (TWAMP) [RFC5357] states that only a Differentiated Services Code Point (DSCP) [RFC2474], [RFC3168], [RFC3260] value 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. The [RFC7750] introduced optional DSCP Monitoring mode that can be negotiated using TWAMP Control protocol and supported by TWAMP-Test protocol or by TWAMP Light mode. Still the TWAMP-Test protocol does not support discovery of how Differentiated Services policies configured along the IP path process various DSCP values in single test session. Hence method defined in [RFC7750] can be characterized as per-session DSCP Monitoring. To provide higher efficiency and flexibility to monitoring how Differentiated Services policies being applied this document proposes ability to control DSCP value to be

used by Session-Reflector for each TWAMP-Test packet. Such method can be characterized as per-packet DSCP monitoring with TWAMP.

This document describes an OPTIONAL feature for TWAMP. It is called the DSCP and ECN Testing. It allows the Session-Sender to use set of DSCP values through single test session and to instruct the Session-Reflector on what DSCP value it must use for the reflected test packet. Furthermore this feature tracks the Explicit Congestion Notification (ECN) [RFC2474], [RFC3168], [RFC3260] 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 Services Code Point

ECN: Explicit Congestion Notification

IPPM: IP Performance Metrics

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 flag to identify the ability of a Session-Reflector to support the new Session-Sender packet format in the TWAMP-Test protocol and to use received DSCP and ECN values in the reflected to a Session-Sender test packet, See the Section 3 for details on the assigned bit position.

2.1. Setting Up Connection to Test DSCP and ECN

The Server sets the DSCP and ECN Testing flag in the Modes field of the Server Greeting message to indicate its capabilities and willingness to monitor them. If the Control-Client agrees to test DSCP and ECN on some or all test sessions invoked with this control connection, it MUST set the DSCP and ECN Testing flag in the Modes field in the Setup Response message.

2.2. TWAMP-Test Extension

Testing of DSCP and ECN requires support by the Session-Sender and changes the test packet format in all the original (unauthenticated, authenticated and encrypted) modes. Testing of DSCP and ECN does not alter the Session-Reflector test packet format but certain considerations must be taken when and if this mode is accepted in combination with Symmetrical Size mode [RFC6038] and/or with DSCP and ECN Monitoring mode [RFC7750].

2.2.1. Session-Sender Packet Format for DSCP and ECN Testing

When the Session-Sender supports DSCP and ECN Testing it constructs the Reflector DSCP and ECN (R-DSCP-ECN) field, presented in Figure 1, for each test packet it sends to Session-Reflector according to the following procedure:

- value of the Reflector DSCP (R-DSCP) field MUST be set to the value that the Session-Reflector MUST use for the reflected test packet;

- value of the Reflector ECN (R-ECN) field MUST be set to the value that the Session-Reflector MAY use for the reflected test packet.

When the Session-Reflector supports DSCP and ECN Testing mode it uses R-DSCP-ECN field of the received test packet to construct the reflected test packet according to the following procedure

- the R-DSCP field MUST be used as six (least-significant) bits of the Differentiated Service field of the reflected test packet;

- the R-ECN field MAY be used as the two bits of the ECN field of the reflected test packet.

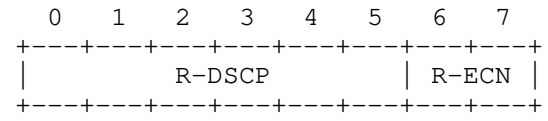


Figure 1: Sender DSCP and ECN field format

Formats of the test packet transmitted by the Session-Sender in unauthenticated, authenticated and encrypted modes been defined in Section 4.1.2 [RFC4656]. For the Session-Sender that supports DSCP and ECN Testing these formats are displayed in Figure 2 and Figure 3.

For unauthenticated mode:

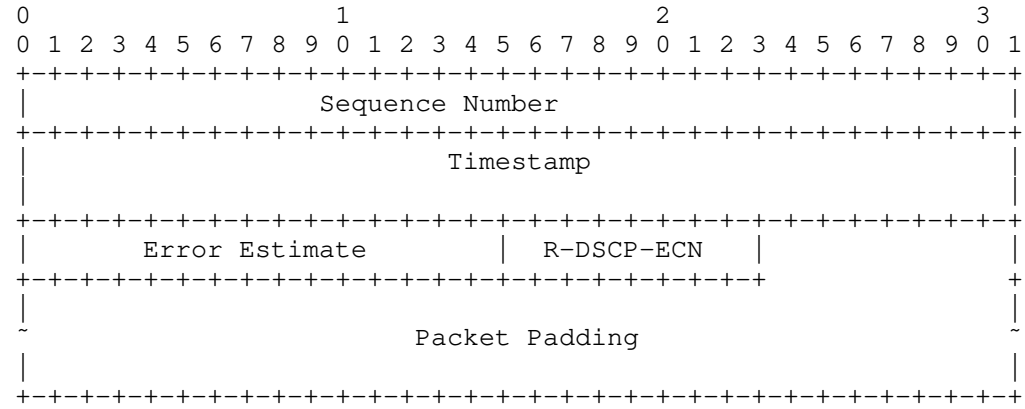


Figure 2: Session-Sender test packet format with DSCP and ECN Testing in unauthenticated mode

Neither Session-Reflector, nor Session-Sender in the DSCP and ECN Testing mode analyze, nor act on ECN value of the received TWAMP test packet and therefore ignore congestion indications from the network. It is expected that sending rates are low enough, as TWAMP deployment experience had demonstrated since TWAMP base RFC 5357 publication in 2008, that ignoring these congestion indications will not significantly contribute to network congestion.

For authenticated and encrypted modes:

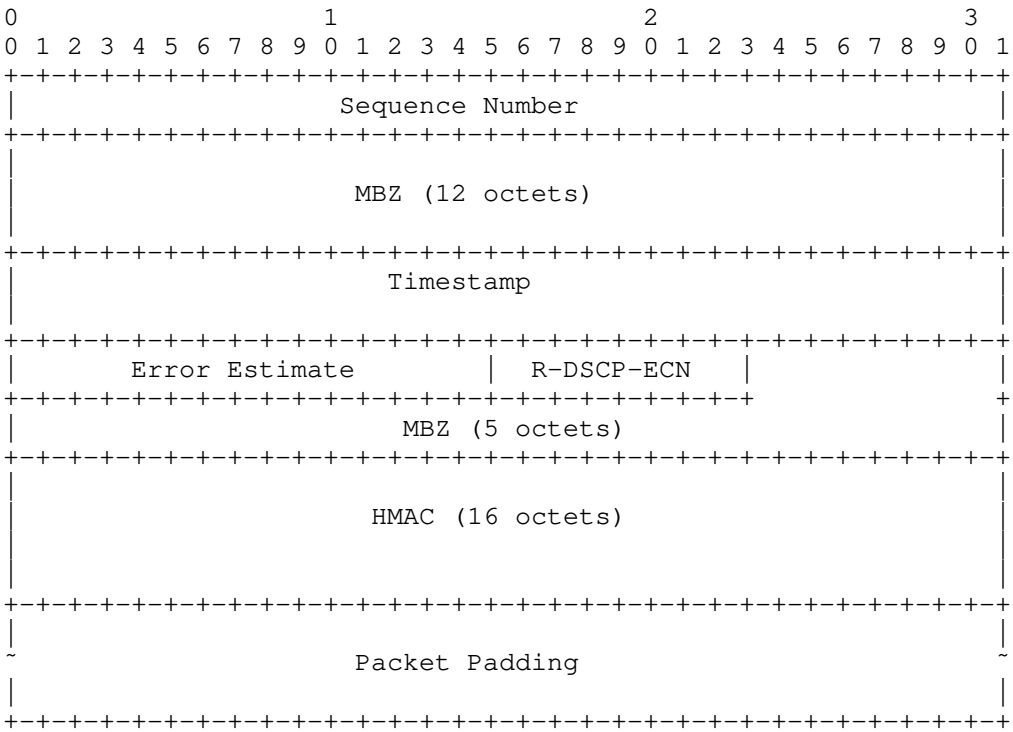


Figure 3: Session-Sender test packet format with DSCP and ECN Testing in authenticated or encrypted modes

2.2.2. Combining DSCP and ECN Testing and Monitoring extensions

[RFC7750] defined DSCP and ECN Monitoring extension. Using testing and monitoring modes in the same test session allows test DSCP in forward and reverse directions because Session-Reflector returns received DSCP and ECN values in S-DSCP-ECN field in the reflected test packet.

2.2.3. DSCP and ECN Testing 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 Testing and Symmetrical Size and/or Reflects Octets modes are being negotiated between Server and Control-Client in Unauthenticated mode, then, because R-DSCP-ECN field increases size of unauthenticated Session-Sender packet by 4 octets, the Padding Length value SHOULD be >= 26 octets to allow for

the truncation process that TWAMP recommends in Section 4.2.1 of [RFC5357].

If DSCP and ECN Testing mode to be used in combination with Symmetrical Size [RFC6038] and DSCP and ECN Monitoring [RFC7750] modes, then the Padding Length value SHOULD be >= 27 octets to allow the truncation process that TWAMP recommends in Section 4.2.1 of [RFC5357].

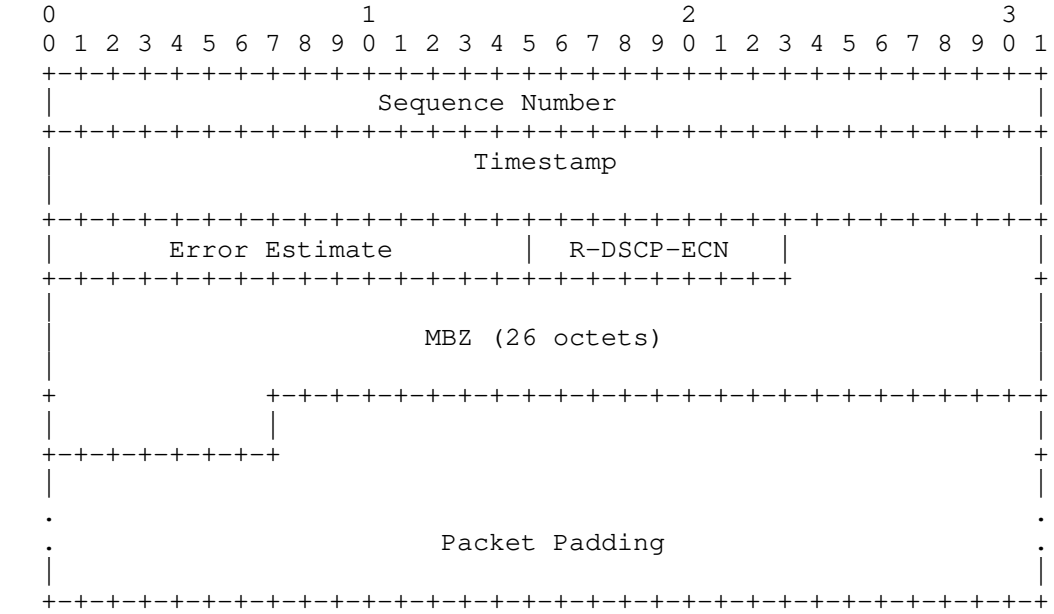


Figure 4: Session-Sender test packet format with DSCP and ECN Testing and Symmetrical Test Packet in unauthenticated mode

2.2.4. 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.

In order to test DSCP over round-trip path between Session-Sender and Session-Reflector it is sufficient that Session-Reflector uses received DSCP value for the refelcted test packet. If the Session-Reflector supports both testing and monitring of DSCP, then TWAMP Light mode MAY be used to test DSCP in forward and rverse directions.

3. IANA Considerations

The TWAMP-Modes registry defined in [RFC5618].

IANA is requested to reserve a new DSCP and ECN Testing Capability as follows:

Bit	Description	Semantics Definition	Reference
TBA	DSCP and ECN Testing Capability	Section 2	This document

Table 1: New Type-P Descriptor Testing Capability

4. Security Considerations

Testing 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]. Sections such as 3.2, 4., 4.1.2, 4.2, and 4.2.1 of [RFC5357] discuss unauthenticated, authenticated, and encrypted modes in varying degrees of detail. The security considerations that apply to any active measurement of live networks are relevant here as well. See the Security Considerations sections in [RFC4656], [RFC5357], and [RFC7750].

5. Acknowledgements

TBD

6. References

6.1. 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>.

[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>.

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

[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>.

[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>.

[RFC7750] Hedin, J., Mirsky, G., and S. Baillargeon, "Differentiated Service Code Point and Explicit Congestion Notification Monitoring in the Two-Way Active Measurement Protocol (TWAMP)", RFC 7750, DOI 10.17487/RFC7750, February 2016, <http://www.rfc-editor.org/info/rfc7750>.

6.2. Informative References

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

Authors' Addresses

Steve Baillargeon
Ericsson

Email: steve.baillargeon@ericsson.com

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.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

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.

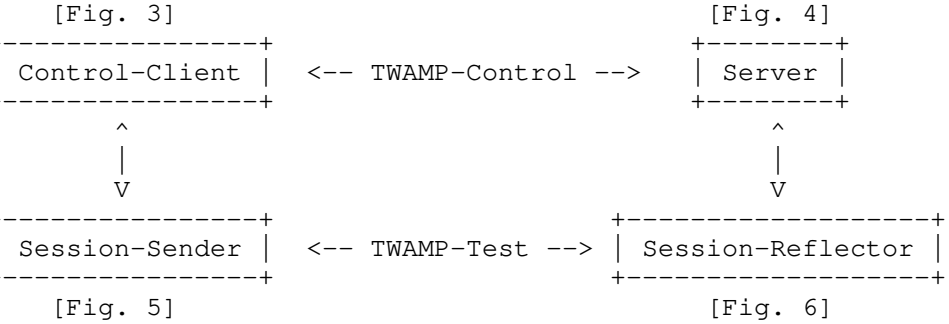


Figure 1: Annotated TWAMP logical model

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.

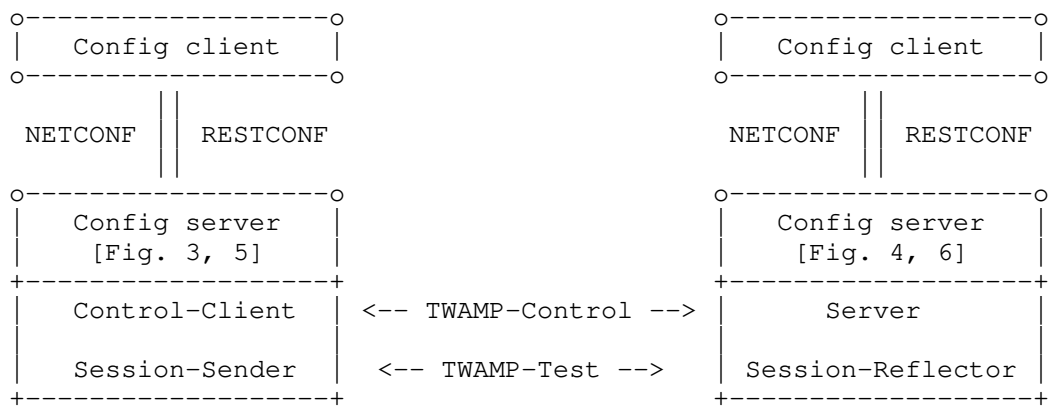


Figure 2: Simplified TWAMP model and protocols

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).

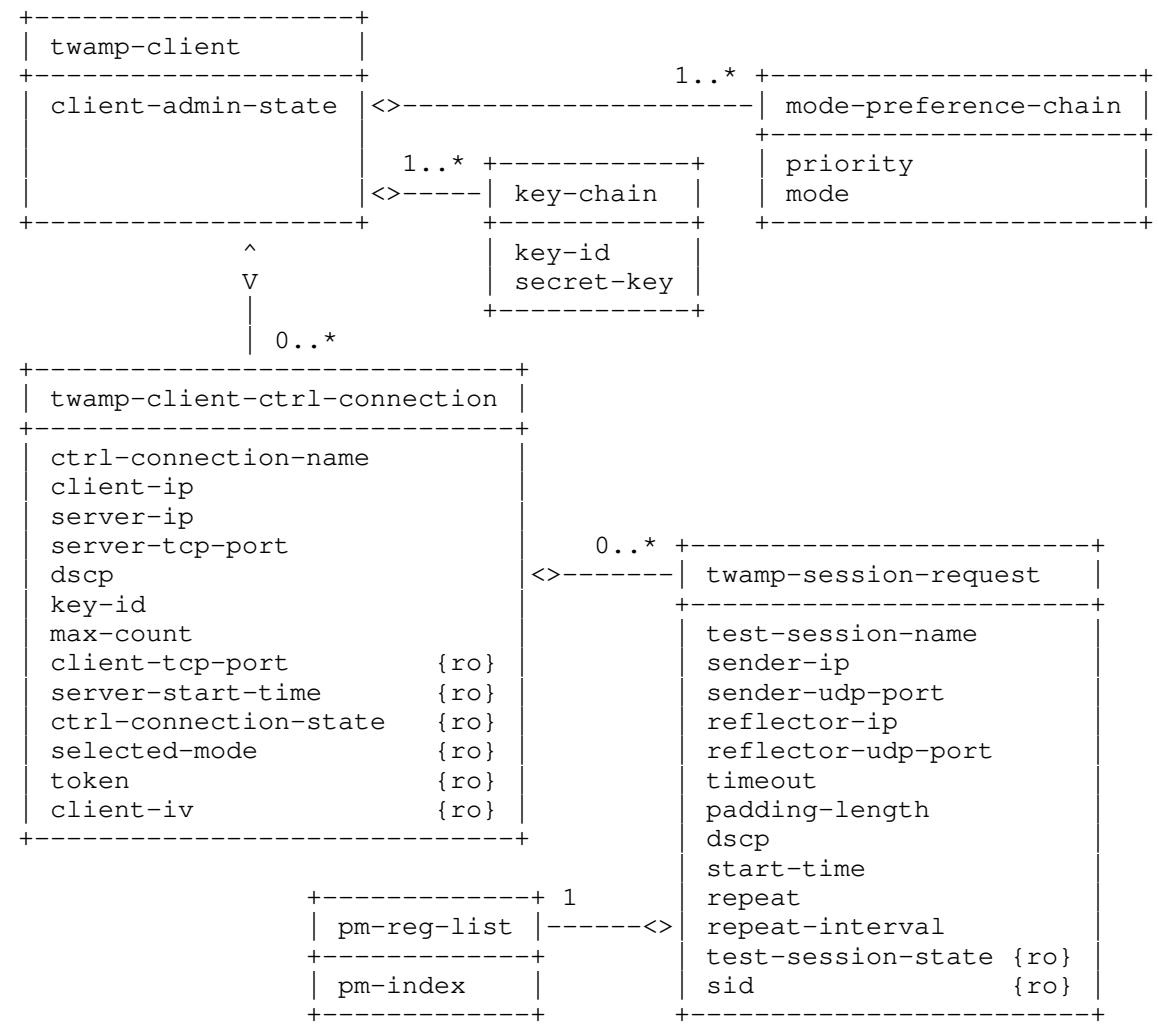


Figure 3: TWAMP Control-Client UML class diagram

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).

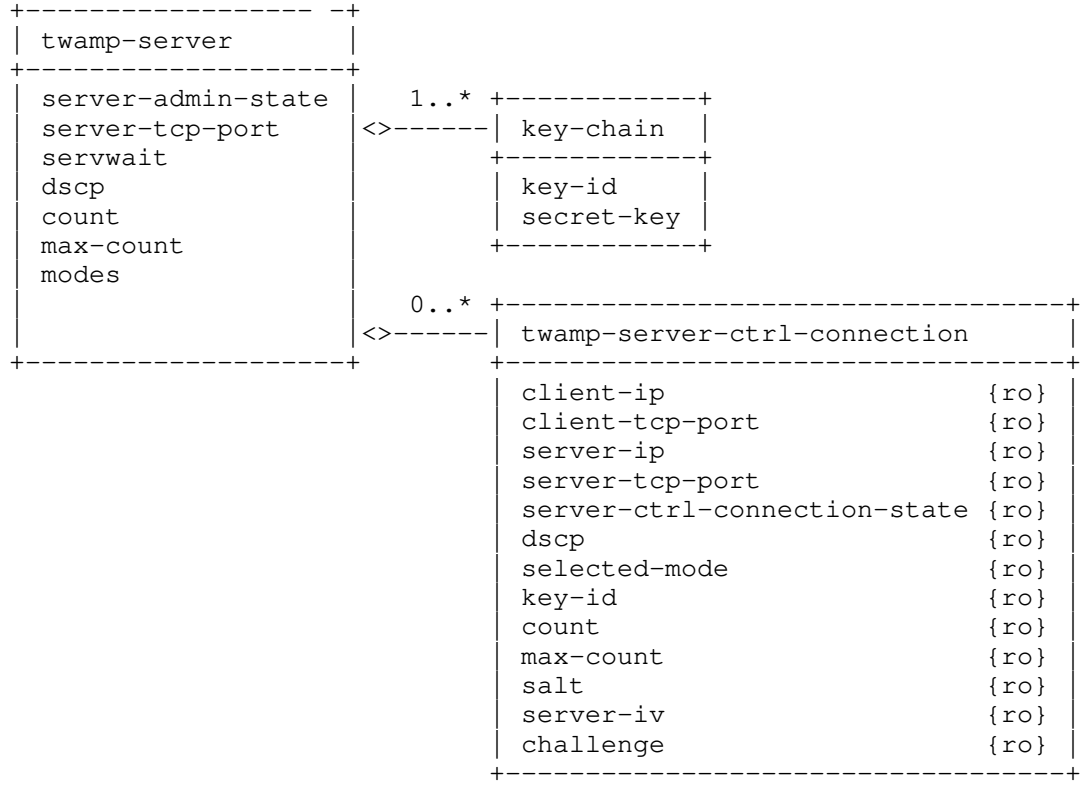


Figure 4: TWAMP Server UML class diagram

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.

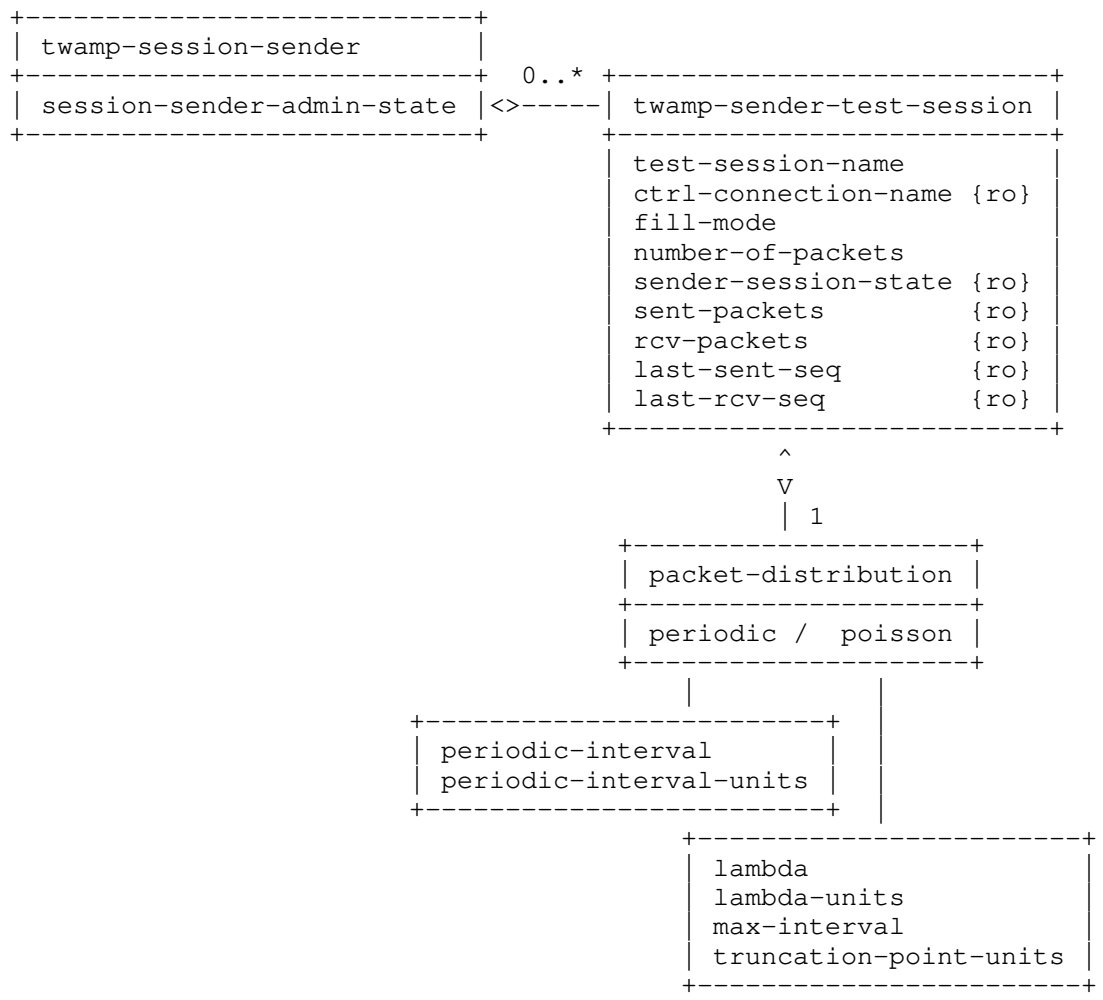


Figure 5: TWAMP Session-Sender UML class diagram

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.

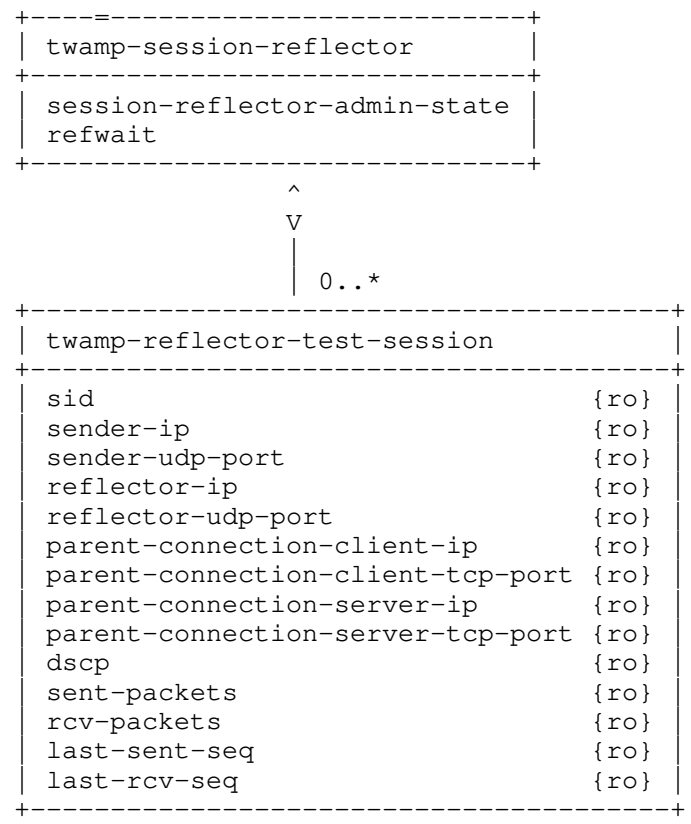


Figure 6: TWAMP Session-Reflector UML class diagram

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: December 28, 2017
June 26, 2017

IPv6 Performance and Diagnostic Metrics (PDM) Destination Option
draft-ietf-ippm-6man-pdm-option-13

Abstract

To assess performance problems, this document describes optional headers embedded in each packet that provide sequence numbers and timing information as a basis for measurements. Such measurements may be interpreted in real-time or after the fact. This document specifies the Performance and Diagnostic Metrics (PDM) destination options extension header. The field limits, calculations, and usage in measurement of PDM are included in this document.

Table of Contents

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.

As defined in RFC2460 [RFC2460], destination options are carried by the IPv6 Destination Options extension header. Destination options include optional information that need be examined only by the IPv6 node given as the destination address in the IPv6 header, not by routers or other "middle boxes". This document specifies the Performance and Diagnostic Metrics (PDM) destination option. The field limits, calculations, and usage in measurement of the PDM destination options extension header are included in this document.

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 Rationale for defined solution

The current IPv6 specification does not provide timing nor a similar field in the IPv6 main header or in any extension header. The IPv6 Performance and Diagnostic Metrics destination option (PDM) provides such fields.

Advantages include:

1. Real measure of actual transactions.

2. Ability to span organizational boundaries with consistent instrumentation.

3. No time synchronization needed between session partners

4. Ability to handle all transport protocols (TCP, UDP, SCTP, etc) in a uniform way

The PDM provides the ability to determine quickly if the (latency) problem is in the network or in the server (application). That is, it is a fast way to do triage. For more information on background and usage of PDM, see Appendix A.

1.3 IPv6 Transition Technologies

In the path to full implementation of IPv6, transition technologies such as translation or tunneling may be employed. It is possible that an IPv6 packet containing PDM may be dropped if using IPv6 transition technologies. For example, an implementation using a translation technique (IPv6 to IPv4) which does not support or recognize the IPv6 Destination Options extension header may simply drop the packet rather than translating it without the extension header.

It is also possible that some devices in the network may not correctly handle multiple IPv6 Extension Headers, including the IPv6 Destination Option. For example, adding the PDM header to a packet may push the layer 4 information to a point in the packet where it is not visible to filtering logic, and may be dropped. This kind of situation is expected to become rare over time.

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 base fields:

```
PSNTP    : Packet Sequence Number This Packet
PSNLR    : Packet Sequence Number Last Received
DELTATLR : Delta Time Last Received
DELTATLS : Delta Time Last Sent
```

Other fields for storing time scaling factors are also in the PDM and will be described in section 3.

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 *Network* delay is the delay for packet transfer from a source host to a destination host and then back to the source 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 needs to 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 implemented as an IPv6 Option carried in the Destination Options Header. The PDM does not require time synchronization.

3.2 Performance and Diagnostic Metrics Destination Option

3.2.1 PDM Layout

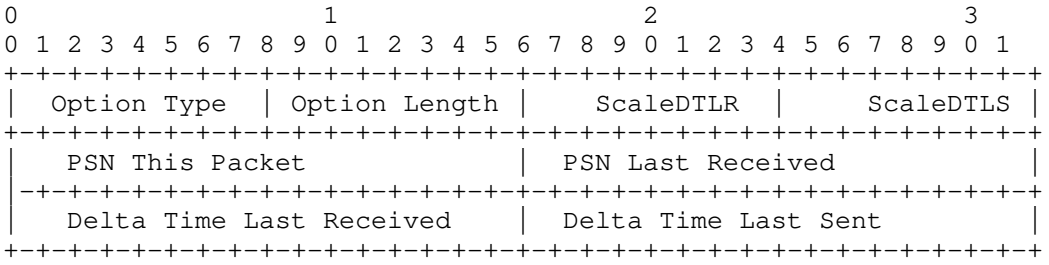
The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) contains the following fields:

```
SCALEDTLR: Scale for Delta Time Last Received
SCALEDTLS: Scale for Delta Time Last Sent
PSNTP    : Packet Sequence Number This Packet
PSNLR    : Packet Sequence Number Last Received
DELTATLR : Delta Time Last Received
DELTATLS : Delta Time Last Sent
```

PDM has alignment requirements. Following the convention in IPv6, these options are aligned in a packet so that multi-octet values

within the Option Data field of each option fall on natural boundaries (i.e., fields of width n octets are placed at an integer multiple of n octets from the start of the header, for n = 1, 2, 4, or 8) [RFC2460].

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]

In keeping with RFC2460[RFC2460], the two high 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.

The third high 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 high order bit MUST be 0.

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 10.

Scale Delta Time Last Received (SCALEDTLR)

8-bit unsigned integer. This is the scaling value for the Delta Time Last Received (DELTATLR) field. The possible values are from 0-255. See Section 4 for further discussion on Timing Considerations and formatting of the scaling values.

Scale Delta Time Last Sent (SCALEDTLS)

8-bit signed integer. This is the scaling value for the Delta Time Last Sent (DELTATLS) field. The possible values are from 0 to 255.

Packet Sequence Number This Packet (PSNTP)

16-bit unsigned integer. This field will wrap. It is intended for use while analyzing packet traces.

Initialized at a random number and incremented monotonically for each packet of the session flow of the 5-tuple. The random number initialization is intended to make it harder to spoof and insert such packets.

Operating systems MUST implement a separate packet sequence number counter per 5-tuple.

Packet Sequence Number Last Received (PSNLR)

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

This field is initialized to 0.

Delta Time Last Received (DELTATLR)

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

Delta Time Last Received = (Send time packet n - Receive time packet n-1)

Delta Time Last Sent (DELTATLS)

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

Delta Time Last Sent = (Receive time packet n - Send time packet n-1)

3.2.2 Base Unit for Time Measurement

A time differential is always a whole number in a CPU; it is the unit specification -- hours, seconds, nanoseconds -- that determine what the numeric value means. For PDM, the base time unit is 1 attosecond (asec). This allows for a common unit and scaling of the time differential among all IP stacks and hardware implementations.

Note that PDM provides the ability to measure both time differentials that are extremely small, and time differentials in a Delay/Disruption Tolerant Networking (DTN) environment where the

delays may be very great. To store a time differential in just 16 bits that must range in this way will require some scaling of the time differential value.

One issue is the conversion from the native time base in the CPU hardware of whatever device is in use to some number of attoseconds. It might seem this will be an astronomical number, but the conversion is straightforward. It involves multiplication by an appropriate power of 10 to change the value into a number of attoseconds. Then, to scale the value so that it fits into DELTATLR or DELTATLS, the value is shifted by of a number of bits, retaining the 16 high-order or most significant bits. The number of bits shifted becomes the scaling factor, stored as SCALEDTLR or SCALEDTLS, respectively. For additional information of this process, including examples, please see Appendix A.

3.3 Header Placement

The PDM Destination Option is placed as defined in RFC2460 [RFC2460]. There may be a choice of where to place the Destination Options header. If using ESP mode, please see section 3.4 of this document for placement of the PDM Destination Options header.

For each IPv6 packet header, the PDM MUST NOT appear more than once. However, an encapsulated packet MAY contain a separate PDM associated with each encapsulated IPv6 header.

3.4 Header Placement Using IPSec ESP Mode

IPSec Encapsulating Security Payload (ESP) is defined in [RFC4303] and is widely used. Section 3.1.1 of [RFC4303] discusses placement of Destination Options Headers.

The placement of PDM is different depending on if ESP is used in tunnel or transport mode.

In ESP case, no 5-tuple is available, as there are no port numbers. ESP flow should be identified only by using SADDR, DADDR and PROTOC. The SPI numbers SHOULD be ignored when considering the flow over which PDM information is measured.

3.4.1 Using ESP Transport Mode

Note that Destination Options may be placed before or after ESP or both. If using PDM in ESP transport mode, PDM MUST be placed after the ESP header so as not to leak information.

3.4.2 Using ESP Tunnel Mode

Note that Destination Options may be placed before or after ESP or both in both the outer set of IP headers and the inner set of IP headers. A tunnel endpoint that creates a new packet may decide to use PDM independent of the use of PDM of the original packet to enable delay measurements between the two tunnel endpoints.

3.5 Implementation Considerations

3.5.1 PDM Activation

An implementation should provide an interface to enable or disable the use of PDM. This specification recommends having PDM off by default.

PDM MUST NOT be turned on merely if a packet is received with a PDM header. The received packet could be spoofed by another device.

3.5.2 PDM Timestamps

The PDM timestamps are intended to isolate wire time from server or host time, but may necessarily attribute some host processing time to network latency.

RFC2330 [RFC2330] "Framework for IP Performance Metrics" describes two notions of wire time in section 10.2. These notions are only defined in terms of an Internet host H observing an Internet link L at a particular location:

+ For a given IP packet P, the 'wire arrival time' of P at H on L is the first time T at which any bit of P has appeared at H's observational position on L.

+ For a given IP packet P, the 'wire exit time' of P at H on L is the first time T at which all the bits of P have appeared at H's observational position on L.

This specification does not define the exact H's observing position on L. That is left for the deployment setups to define. However, the position where PDM timestamps are taken SHOULD be as close to the physical network interface as possible. Not all implementations will be able to achieve the ideal level of measurement.

3.6 Dynamic Configuration Options

If the PDM destination options extension header is used, then it MAY be turned on for all packets flowing through the host, applied to an upper-layer protocol (TCP, UDP, SCTP, etc), a local port, or IP address only. These are at the discretion of the implementation.

3.7 Information Access and Storage

Measurement information provided by PDM may be made accessible for higher layers or the user itself. Similar to activating the use of PDM, the implementation may also provide an interface to indicate if received

PDM information may be stored, if desired. If a packet with PDM information is received and the information should be stored, the upper layers may be notified. Furthermore, the implementation should define a configurable maximum lifetime after which the information can be removed as well as a configurable maximum amount of memory that should be allocated for PDM information.

4 Security Considerations

PDM may introduce some new security weaknesses.

4.1 Resource Consumption and Resource Consumption Attacks

PDM needs to calculate the deltas for time and keep track of the sequence numbers. This means that control blocks which reside in memory may be kept at the end hosts per 5-tuple.

A limit on how much memory is being used SHOULD be implemented.

Without a memory limit, any time a control block is kept in memory, an attacker can try to misuse the control blocks to cause excessive resource consumption. This could be used to compromise the end host.

PDM is used only at the end hosts and memory is used only at the end host and not at routers or middle boxes.

4.2 Pervasive monitoring

Since PDM passes in the clear, a concern arises as to whether the data can be used to fingerprint the system or somehow obtain information about the contents of the payload.

Let us discuss fingerprinting of the end host first. It is possible that seeing the pattern of deltas or the absolute values could give some information as to the speed of the end host - that is, if it is a very fast system or an older, slow device. This may be useful to

the attacker. However, if the attacker has access to PDM, the attacker also has access to the entire packet and could make such a deduction based merely on the time frames elapsed between packets WITHOUT PDM.

As far as deducing the content of the payload, in terms of the application level information such as web page, user name, user password and so on, it appears to us that PDM is quite unhelpful in this regard. Having said that, the ability to separate wire-time from processing time may potentially provide an attacker with additional information. It is conceivable that an attacker could attempt to deduce the type of application in use by noting the server time and payload length. Some encryption algorithms attempt to obfuscate the packet length to avoid just such vulnerabilities. In the future, encryption algorithms may wish to obfuscate the server time as well.

4.3 PDM as a Covert Channel

PDM provides a set of fields in the packet which could be used to leak data. But, there is no real reason to suspect that PDM would be chosen rather than another part of the payload or another Extension Header.

A firewall or another device could sanity check the fields within the PDM but randomly assigned sequence numbers and delta times might be expected to vary widely. The biggest problem though is how an attacker would get access to PDM in the first place to leak data. The attacker would have to either compromise the end host or have Man in the Middle (MitM). It is possible that either one could change the fields. But, then the other end host would get sequence numbers and deltas that don’t make any sense.

It is conceivable that someone could compromise an end host and make it start sending packets with PDM without the knowledge of the host. But, again, the bigger problem is the compromise of the end host. Once that is done, the attacker probably has better ways to leak data.

Having said that, if a PDM aware middle box or an implementation (destination host) detects some number of "nonsensical" sequence numbers or timing information, it could take action to block, discard, or alert on this traffic.

4.4 Timing Attacks

The fact that PDM can help in the separation of node processing time

from network latency brings value to performance monitoring. Yet, it is this very characteristic of PDM which may be misused to make certain new type of timing attacks against protocols and implementations possible.

Depending on the nature of the cryptographic protocol used, it may be possible to leak the credentials of the device. For example, if an attacker can see that PDM is being used, then the attacker might use PDM to launch a timing attack against the keying material used by the cryptographic protocol.

An implementation may want to be sure that PDM is enabled only for certain ip addresses, or only for some ports. Additionally, the implementation SHOULD require an explicit restart of monitoring after a certain time period (for example for 1 hour), to make sure that PDM is not accidentally left on after debugging has been done etc.

Even so, if using PDM, a user "Consent to be Measured" SHOULD be a pre-requisite for using PDM. Consent is common in enterprises and with some subscription services. The actual content of "Consent to be Measured" will differ by site but it SHOULD make clear that the traffic is being measured for quality of service and to assist in diagnostics as well as to make clear that there may be potential risks of certain vulnerabilities if the traffic is captured during a diagnostic session.

5 IANA Considerations

This draft requests an Destination Option Type assignment with the act bits set to 00 and the chg bit set to 0 from the Destination Options and Hop-by-Hop Options sub-registry of Internet Protocol Version 6 (IPv6) Parameters [ref to RFCs and URL below].

http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2

Hex Value	Binary Value act chg rest	Description	Reference
TBD	TBD	Performance and Diagnostic Metrics (PDM)	[This draft]

6 References

6.1 Normative References

[RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989.

[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.

[RFC4303] Kent, S, "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

6.2 Informative References

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

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

Appendix A: Context for PDM

A.1 End User Quality of Service (QoS)

The timing values in the PDM embedded in 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.

Low, reliable and acceptable response times are not just "nice to have". On many networks, the impact can be financial hardship or can endanger human life. In some cities, the emergency police contact

system operates over IP; law enforcement, at all levels, use 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 simple solution to support response time measurements.

A.2 Need for a Packet Sequence Number (PSN)

While performing network diagnostics of an end-to-end connection, it often becomes necessary to isolate the factors along the network path responsible for problems. Diagnostic data may be collected at multiple places along the path (if possible), or at the source and destination. Then, in post-collection processing, 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).

A.3 Rationale for Defined Solution

One of the important functions of PDM is to allow you to quickly dispatch the right set of diagnosticians. Within network or server latency, there may be many components. The job of the diagnostician is to rule each one out until the culprit is found.

How PDM fits into this diagnostic picture is that PDM will quickly tell you how to escalate. PDM will point to either the network area or the server area. Within the server latency, PDM does not tell you if the bottleneck is in the IP stack or the application or buffer allocation. Within the network latency, PDM does not tell you which of the network segments or middle boxes is at fault.

What PDM does tell you is whether the problem is in the network or the server.

A.4 Use PDM with Other Headers

For diagnostics, one my want to use PDM with other headers (L2, L3, etc). For example, if PDM is 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 Appendix C.

Appendix B : Timing Considerations

B.1 Timing Differential Calculations

The time counter in a CPU is a binary whole number, representing a number of milliseconds (msec), microseconds (usec) or even picoseconds (psec). Representing one of these values as attoseconds (asec) means multiplying by 10 raised to some exponent. Refer to this table of equalities:

Base value	= # of sec	= # of asec	1000s of asec
1 second	1 sec	10**18 asec	1000**6 asec
1 millisecond	10**-3 sec	10**15 asec	1000**5 asec
1 microsecond	10**-6 sec	10**12 asec	1000**4 asec
1 nanosecond	10**-9 sec	10**9 asec	1000**3 asec
1 picosecond	10**-12 sec	10**6 asec	1000**2 asec
1 femtosecond	10**-15 sec	10**3 asec	1000**1 asec

For example, if you have a time differential expressed in microseconds, since each microsecond is 10**12 asec, you would multiply your time value by 10**12 to obtain the number of attoseconds. If you time differential is expressed in nanoseconds, you would multiply by 10**9 to get the number of attoseconds.

The result is a binary value that will need to be shortened by a number of bits so it will fit into the 16-bit PDM DELTA field.

The next step is to divide by 2 until the value is contained in just 16 significant bits. The exponent of the value in the last column of of the table is useful here; the initial scaling factor is that exponent multiplied by 10. This is the minimum number of low-order bits to be shifted-out or discarded. It represents dividing the time value by 1024 raised to that exponent.

The resulting value may still be too large to fit into 16 bits, but can be normalized by shifting out more bits (dividing by 2) until the value fits into the 16-bit DELTA field. The number of extra bits shifted out is then added to the scaling factor. The scaling factor, the total number of low-order bits dropped, is the SCALEDTL value.

For example: say an application has these start and finish timer values (hexadecimal values, in microseconds):

Finish:	27C849234 usec	(02:57:58.997556)
-Start:	27C83F696 usec	(02:57:58.957718)
==========	=========	==========
Difference	9B9E usec	00.039838 sec or 39838 usec

To convert this differential value to binary attoseconds, multiply the number of microseconds by 10**12. Divide by 1024**4, or simply discard 40 bits from the right. The result is 36232, or 8D88 in hex, with a scaling factor or SCALEDTL value of 40.

For another example, presume the time differential is larger, say 32.311072 seconds, which is 32311072 usec. Each microsecond is 10**12 asec, so multiply by 10**12, giving the hexadecimal value 1C067FCCAE8120000. Using the initial scaling factor of 40, drop the last 10 characters (40 bits) from that string, giving 1C067FC. This will not fit into a DELTA field, as it is 25 bits long. Shifting the value to the right another 9 bits results in a DELTA value of E033, with a resulting scaling factor of 49.

When the time differential value is a small number, regardless of the time unit, the exponent trick given above is not useful in determining the proper scaling value. For example, if the time differential is 3 seconds and you want to convert that directly, you would follow this path:

```
3 seconds = 3*10**18 asec (decimal)
          = 29A2241AF62C0000 asec (hexadecimal)
```

If you just truncate the last 60 bits, you end up with a delta value of 2 and a scaling factor of 60, when what you really wanted was a delta value with more significant digits. The most precision with which you can store this value in 16 bits is A688, with a scaling factor of 46.

B.2 Considerations of this time-differential representation

There are a few considerations to be taken into account with this representation of a time differential. The first is whether there are any limitations on the maximum or minimum time differential that can be expressed using method of a delta value and a scaling factor. The second is the amount of imprecision introduced by this method.

B.2.1 Limitations with this encoding method

The DELTATLS and DELTATLR fields store only the 16 most-significant bits of the time differential value. Thus the range, excluding the scaling factor, is from 0 to 65535, or a maximum of 2**16-1. This method is further described in [TRAM-TCPM].

The actual magnitude of the time differential is determined by the scaling factor. SCALEDTLR and SCALEDTLS are 8-bit unsigned integers, so the scaling factor ranges from 0 to 255. The smallest number that can be represented would have a value of 1 in the delta field and a

value of 0 in the associated scale field. This is the representation for 1 attosecond. Clearly this allows PDM to measure extremely small time differentials.

On the other end of the scale, the maximum delta value is 65535, or FFFF in hexadecimal. If the maximum scale value of 255 is used, the time differential represented is 65535*2**255, which is over 3*10**55 years, essentially, forever. So there appears to be no real limitation to the time differential that can be represented.

B.2.2 Loss of precision induced by timer value truncation

As PDM specifies the DELTATLR and DELTATLS values as 16-bit unsigned integers, any time the precision is greater than those 16 bits, there will be truncation of the trailing bits, with an accompanying loss of precision in the value.

Any time differential value smaller than 65536 asec can be stored exactly in DELTATLR or DELTATLS, because the representation of this value requires at most 16 bits.

Since the time differential values in PDM are measured in attoseconds, the range of values that would be truncated to the same encoded value is 2**(Scale)-1 asec.

For example, the smallest time differential that would be truncated to fit into a delta field is

1 0000 0000 0000 0000 = 65536 asec

This value would be encoded as a delta value of 8000 (hexadecimal) with a scaling factor of 1. The value

1 0000 0000 0000 0001 = 65537 asec

would also be encoded as a delta value of 8000 with a scaling factor of 1. This actually is the largest value that would be truncated to that same encoded value. When the scale value is 1, the value range is calculated as 2**1 - 1, or 1 asec, which you can see is the difference between these minimum and maximum values.

The scaling factor is defined as the number of low-order bits truncated to reduce the size of the resulting value so it fits into a 16-bit delta field. If, for example, you had to truncate 12 bits, the loss of precision would depend on the bits you truncated. The range of these values would be

0000 0000 0000 = 0 asec

to
1111 1111 1111 = 4095 asec

So the minimum loss of precision would be 0 asec, where the delta value exactly represents the time differential, and the maximum loss of precision would be 4095 asec. As stated above, the scaling factor of 12 means the maximum loss of precision is 2**12-1 asec, or 4095 asec.

Compare this loss of precision to the actual time differential. The range of actual time differential values that would incur this loss of precision is from

1000 0000 0000 0000 0000 0000 0000 = 2**27 asec or 134217728 asec
to
1111 1111 1111 1111 1111 1111 1111 = 2**28-1 asec or 268435455 asec

Granted, these are small values, but the point is, any value between these two values will have a maximum loss of precision of 4095 asec, or about 0.00305% for the first value, as encoded, and at most 0.001526% for the second. These maximum-loss percentages are consistent for all scaling values.

Appendix C: Sample Packet Flows

C.1 PDM Flow - Simple Client Server

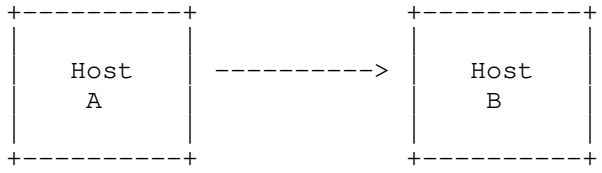
Following is a sample simple flow for the PDM with one packet sent from Host A and one packet received by Host B. The PDM does not require time synchronization between Host A and Host B. The calculations to derive meaningful metrics for network diagnostics are shown below each packet sent or received.

C.1.1 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:   -
DELTATLR : Delta Time Last Received:               -
SCALEDTLR: Scale of Delta Time Last Received:      0
DELTATLS : Delta Time Last Sent:                   -
SCALEDTLS: Scale of Delta Time Last Sent:          0
```

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 time in these examples is shown in seconds for the sake of simplicity.

C.1.2 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.

C.1.3 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

The result of this calculation is called: Delta Time Last Received (DELTATLR)

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:

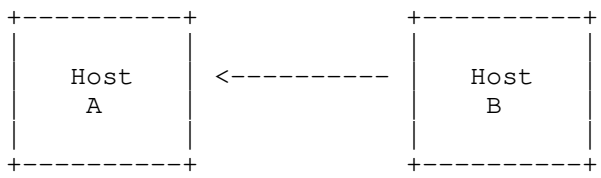
Packet Sequence Number of the last packet received:	25
Time the last packet was received:	11:00:03
Packet Sequence Number of this packet:	12
Time this packet is being sent:	11:00:07

Now a delta value to be sent out in the packet can be calculated. DELTATLR becomes:

4 seconds = 11:00:07 - 11:00:03 = 3782DACE9D900000 asec

This is the derived metric: Server Delay. The time and scaling factor must be converted; in this case, the time differential is DE0B, and the scaling factor is 2E, or 46 in decimal. Then, these values, 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
DELTATLR : Delta Time Last Received:              DE0B (4 seconds)
SCALEDTLR: Scale of Delta Time Last Received:     2E (46 decimal)
DELTATLS : Delta Time Last Sent:                   -
SCALEDTLS: Scale of Delta Time Last Sent:          0
```

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

C.1.4 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 time may also be called total Overall Round-Trip Time (RTT) which includes Network RTT and Host Response Time.

This derived metric we will call Delta Time Last Sent (DELTATLS)

Round trip delay can now be calculated. The formula is:

Round trip delay = (Delta Time Last Sent - Delta Time Last Received)

Or:

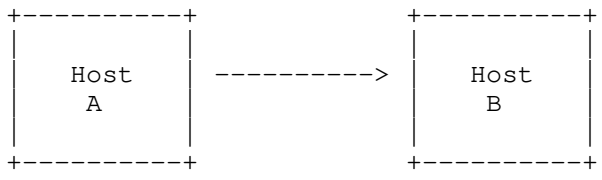
Round trip delay = 12 - 4 or 8

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.

Note: this simple example assumes one send and one receive. That is done only for purposes of explaining the function of the PDM. In cases where there are multiple packets returned, one would take the time in the last packet in the sequence. The calculations of such timings and intelligent processing is the function of post-processing of the data.

C.1.5 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
DELTATLR : Delta Time Last Received:               0
SCALEDTLS: Scale of Delta Time Last Received       0
DELTATLS : Delta Time Last Sent:                 A688 (scaled value)
SCALEDTLR: Scale of Delta Time Last Received:     30 (48 decimal)
```

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

C.2 Other Flows

What has been 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.

C.2.1 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:

Note: The time is expressed in generic units for simplicity. That is, these values do not represent a number of attoseconds, microseconds or any other real units of time.

Packet	Sender	PSN This Packet	PSN Last Recvd	Delta Time Last Recvd	Delta Time 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 Time Last Sent will be seen as used. Recall, Delta Time 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 time units from the server but takes 10 times that long 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.

C.2.2 PDM Flow - Multiple Send Traffic

Assume that two packets are sent for each ACK from the server. For example, a TCP flow will do this, per RFC1122 [RFC1122] Section-4.2.3.

Packet	Sender	PSN This Packet	PSN Last Recvd	Delta Time Last Recvd	Delta Time Last Sent
1	Server	1	0	0	0
2	Server	2	0	0	5
3	Client	1	2	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 Time Last received of 20. Recall that Delta Time Last Received is the Send time of packet 3 - receive time of packet 2. So, what does one know now? In this case, Delta Time 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 such information is now available in a uniform manner for all applications and all protocols without

extensive changes required to applications.

C.2.3 PDM Flow - Multiple Send with Errors

Let us now look at a case of how PDM may be able to help in a case of TCP retransmission and add to the information that is sent in the TCP header.

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 Time LastRecvd	Delta Time 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 receive 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 Time LastRecvd	Delta Time 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 Time LastRecvd	Delta Time 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 is lost again. 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 Time LastRecvd	Delta Time LastSent	TCP SEQ	Data Bytes
1	Server	5	0	0	60	223	100

If now, this packet arrives at the destination, one has a very good idea that packets exist which are being sent from the server as retransmissions and not arriving at 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. 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.

Appendix D: Potential Overhead Considerations

One might wonder 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 for various assumed RTTs.

Bytes in Packet	RTT	Bytes Per Millisec	Bytes in PDM	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.

Bytes in Packet	RTT	Bytes Per Millisec	Bytes in PDM	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.

Bytes in Packet	RTT	Bytes Per Microsec	Bytes in PDM	New RTT	Overhead
1000	20 micro	50	16	20.320	.320 micro

As with other diagnostic tools, such as packet traces, a certain amount of processing time will be required to create and process PDM. Since PDM is lightweight (has only a few variables), we expect the processing time to be minimal.

Acknowledgments

The authors would like to thank Keven Haining, Al Morton, Brian Trammel, David Boyes, Bill Jouris, Richard Scheffenegger, and Rick Troth for their comments and assistance. We would also like to thank Tero Kivinen and Jouni Korhonen for their detailed and perceptive reviews.

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 M. 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@bcbsm.com
http://www.bcbsm.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 29, 2019

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

March 28, 2019

Initial Performance Metrics Registry Entries
draft-ietf-ippm-initial-registry-11

Abstract

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

Requirements Language

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

Status of This Memo

Table of Contents

1. Introduction

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

Although there are several standard templates for organizing specifications of performance metrics (see [RFC2679] for an example of the traditional IPPM template, based to large extent on the Benchmarking Methodology Working Group's traditional template in [RFC1242], and see [RFC6390] for a similar template), none of these templates were intended to become the basis for the columns of an IETF-wide registry of metrics. While examining aspects of metric

specifications which need to be registered, it became clear that none of the existing metric templates fully satisfies the particular needs of a registry.

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

2. Scope

This document defines the initial set of Performance Metrics Registry entries, for which IETF approval (following development in the IP Performance Metrics (IPPM) Working Group) will satisfy the requirement for Expert Review. Most are Active Performance Metrics, which are based on RFCs prepared in the IPPM working group of the IETF, according to their framework [RFC2330] and its updates.

3. Registry Categories and Columns

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

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

Registry Categories and Columns, shown as

	Category	
	Column	Column

Summary

Identifier	Name	URIs	Desc.	Reference	Change Controller	Ver

Metric Definition

Reference Definition	Fixed Parameters

Method of Measurement

Reference Method	Packet Stream Generation	Traffic Filter	Sampling Distribution	Run-time Parameters	Role

Output

Type	Reference Definition	Units	Calibration

Administrative Information

Status	Request	Rev	Rev.Date

Comments and Remarks

4. UDP Round-trip Latency and Loss Registry Entries

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

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

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

4.1. Summary

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

4.1.1. ID (Identifier)

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

4.1.2. Name

RTDelay_Active_IP-UDP-Periodic_RFCXXXXsecY_Seconds_95Percentile

RTLoss_Active_IP-UDP-Periodic_RFCXXXXsecY_Percent_LossRatio

4.1.3. URIs

URL: http://<TBD by IANA>/<name>

4.1.4. Description

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

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

4.1.5. Change Controller

IETF

4.1.6. Version (of Registry Format)

1.0

4.2. Metric Definition

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

4.2.1. Reference Definition

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

[RFC2681]

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

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

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

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

[RFC6673]

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

4.2.2. Fixed Parameters

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

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

* DSCP: set to 0

* Hop Count: set to 255

* Protocol: Set to 17 (UDP)

- UDP header values:

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

- UDP Payload

  * total of 100 bytes

Other measurement parameters:

- Tmax: a loss threshold waiting time

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

4.3. Method of Measurement

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

4.3.1. Reference Method

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

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

The calculations on the delay (RTT) SHALL be performed on the conditional distribution, conditioned on successful packet arrival within Tmax. Also, when all packet delays are stored, the process

which calculates the RTT value MAY enforce the Tmax threshold on stored values before calculations. See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

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

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

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

4.3.2. Packet Stream Generation

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

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

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

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

T0 the actual start time of the periodic stream, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]).

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

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

4.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

NA

4.3.4. Sampling Distribution

NA

4.3.5. Run-time Parameters and Data Format

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

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

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

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

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

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

4.3.6. Roles

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

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

4.4. Output

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

4.4.1. Type

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

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

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

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

4.4.2. Reference Definition

For all outputs ---

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

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

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

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

For

RTDelay_Active_IP-UDP-Periodic_RFCXXXXsecY_Seconds_95Percentile:

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

For

RTLoss_Active_IP-UDP-Periodic_RFCXXXXsecY_Percent_LossRatio:

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

4.4.3. Metric Units

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

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

4.4.4. Calibration

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

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

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

4.5. Administrative items

4.5.1. Status

Current

4.5.2. Requestor

This RFC numner

4.5.3. Revision

1.0

4.5.4. Revision Date

YYYY-MM-DD

4.6. Comments and Remarks

None.

5. Packet Delay Variation Registry Entry

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

Note: If each Registry entry should only produce a "raw" output or a statistical summary, then the "Output" Category can be split and this section can become two closely-related metrics.

5.1. Summary

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

5.1.1. ID (Identifier)

<insert numeric identifier, an integer>

5.1.2. Name

OWPDV_Active_IP-UDP-Periodic_RFCXXXXsecY_Seconds_95Percentile

5.1.3. URIs

URL: http://<TBD by IANA>/<name>

5.1.4. Description

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

5.1.5. Change Controller

IETF

5.1.6. Version (of Registry Format)

1.0

5.2. Metric Definition

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

5.2.1. Reference Definition

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

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

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

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

See sections 2.4 and 3.4 of [RFC3393]. Singleton delay differences measured are referred to by the variable name "ddT" (applicable to all forms of delay variation). However, this metric entry specifies

the PDV form defined in section 4.2 of [RFC5481], where the singleton PDV for packet i is referred to by the variable name "PDV(i)".

5.2.2. Fixed Parameters

- IPv4 header values:
  - DSCP: set to 0
  - TTL: set to 255
  - Protocol: Set to 17 (UDP)
- IPv6 header values:
  - DSCP: set to 0
  - Hop Count: set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum MUST be calculated and included in the header
- UDP Payload
  - total of 200 bytes

Other measurement parameters:

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

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

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

5.3. Method of Measurement

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

5.3.1. Reference Method

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

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

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

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

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

5.3.2. Packet Stream Generation

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

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

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

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

T0 the actual start time of the periodic stream, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]).

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

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

5.3.3. Traffic Filtering (observation) Details

NA

5.3.4. Sampling Distribution

NA

5.3.5. Run-time Parameters and Data Format

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

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

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

[RFC2330]. When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval. The start time is controlled through other means.

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

5.3.6. Roles

5.4. Output

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

5.4.1. Type

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

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

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

5.4.2. Reference Definition

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

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

95Percentile The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction

digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

5.4.3. Metric Units

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

5.4.4. Calibration

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

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

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

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

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

5.5. Administrative items

5.5.1. Status

Current

5.5.2. Requestor

This RFC number

5.5.3. Revision

1.0

5.5.4. Revision Date

YYYY-MM-DD

5.6. Comments and Remarks

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

6. DNS Response Latency and Loss Registry Entries

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

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

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

6.1. Summary

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

6.1.1. ID (Identifier)

<insert numeric identifier, an integer>

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

6.1.2. Name

RTDNS_Active_IP-UDP-Poisson_RFCXXXXsecY_Seconds_Raw

RLDNS_Active_IP-UDP-Poisson_RFCXXXXsecY_Logical_Raw

6.1.3. URI

URI: Prefix urn:ietf:metrics:perf:<name>

URL: http://<TBD by IANA>/<name>

6.1.4. Description

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

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

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

6.1.5. Change Controller

IETF

6.1.6. Version (of Registry Format)

1.0

6.2. Metric Definition

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

6.2.1. Reference Definition

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

[RFC1035]

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

[RFC2681]

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

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

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

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

[RFC6673]

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

6.2.2. Fixed Parameters

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

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255

    * Protocol: Set to 17 (UDP)

  o IPv6 header values:

    * DSCP: set to 0

    * Hop Count: set to 255

    * Protocol: Set to 17 (UDP)

  o UDP header values:

    * Source port: 53

    * Destination port: 53

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

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

    * The DNS header section contains:

      + Identification (see the Run-time column)

      + QR: set to 0 (Query)

      + OPCODE: set to 0 (standard query)

      + AA: not set

      + TC: not set

      + RD: set to one (recursion desired)

      + RA: not set

      + RCODE: not set

      + QDCOUNT: set to one (only one entry)

      + ANCOUNT: not set

      + NSCOUNT: not set

      + ARCOUNT: not set

* The Question section contains:
  + QNAME: the Fully Qualified Domain Name (FQDN) provided as input for the test, see the Run-time column
  + QTYPE: the query type provided as input for the test, see the Run-time column
  + QCLASS: set to 1 for IN
* The other sections do not contain any Resource Records.

Other measurement parameters:

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

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

6.3. Method of Measurement

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

6.3.1. Reference Method

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

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

The calculations on the delay (RTT) SHALL be performed on the conditional distribution, conditioned on successful packet arrival within Tmax. Also, when all packet delays are stored, the process

which calculates the RTT value MAY enforce the Tmax threshold on stored values before calculations. See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

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

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

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

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

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

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

6.3.2. Packet Stream Generation

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

Section 11.1.3 of RFC 2681 [RFC2330] provides three methods to generate Poisson sampling intervals. The reciprocal of lambda is the

average packet rate, thus the Run-time Parameter is Reciprocal_lambda = 1/lambda, in seconds.

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

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

6.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

NA

6.3.4. Sampling Distribution

NA

6.3.5. Run-time Parameters and Data Format

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

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

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

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

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

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

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

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

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

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

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

6.3.6. Roles

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

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

6.4. Output

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

6.4.1. Type

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

6.4.2. Reference Definition

For all outputs:

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

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

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

6.4.3. Metric Units

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

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

6.4.4. Calibration

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

When a measurement controller requests a calibration measurement, the loopback is applied and the result is output in the same format as a

normal measurement with additional indication that it is a calibration result.

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

6.5. Administrative items

6.5.1. Status

Current

6.5.2. Requestor

This RFC number

6.5.3. Revision

1.0

6.5.4. Revision Date

YYYY-MM-DD

6.6. Comments and Remarks

Additional (Informational) details for this entry

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

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

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

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

7.1. Summary

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

7.1.1. ID (Identifier)

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

7.1.2. Name

OWDelay_Active_IP-UDP-Poisson-
Payload250B_RFCXXXXsecY_Seconds_<statistic>

where <statistic> is one of:

- 95Percentile
- Mean
- Min
- Max
- StdDev

OWLoss_Active_IP-UDP-Poisson-
Payload250B_RFCXXXXsecY_Percent_LossRatio

7.1.3. URI and URL

URL: http:\\www.iana.org\ ... <name>

7.1.4. Description

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

where <statistic> is one of:

- 95Percentile
- Mean
- Min

- Max
- StdDev

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

7.2. Metric Definition

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

7.2.1. Reference Definition

For Delay:

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

[RFC7679]

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

[RFC6049]

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

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

For loss:

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

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

7.2.2. Fixed Parameters

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL: set to 255
  - Protocol: Set to 17 (UDP)
- IPv6 header values:
  - DSCP: set to 0
  - Hop Count: set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum MUST be calculated and included in the header
- UDP Payload: TWAMP Test Packet Formats, Section 4.1.2 of [RFC5357]
  - Security features in use influence the number of Padding octets.
  - 250 octets total, including the TWAMP format

Other measurement parameters:

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

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

7.3. Method of Measurement

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

7.3.1. Reference Method

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

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

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

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

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

7.3.2. Packet Stream Generation

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

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

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

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

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

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

7.3.3. Traffic Filtering (observation) Details

NA

7.3.4. Sampling Distribution

NA

7.3.5. Run-time Parameters and Data Format

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

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

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

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

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

7.3.6. Roles

<lists the names of the different roles from the measurement method>

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

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

7.4. Output

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

7.4.1. Type

See subsection titles below for Types.

7.4.2. Reference Definition

For all output types ---

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

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

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

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

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

7.4.2.1. Percentile95

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

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

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

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

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

7.4.2.2. Mean

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

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

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

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

7.4.2.3. Min

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

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

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

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

7.4.2.4. Max

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

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

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

```
Max = (FiniteDelay [j])

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

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

seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

7.4.2.5. Std_Dev

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

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

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

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

7.4.3. Metric Units

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

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

7.4.4. Calibration

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

For one-way delay measurements, the error calibration must include an assessment of the internal clock synchronization with its external reference (this internal clock is supplying timestamps for measurement). In practice, the time offsets of clocks at both the source and destination are needed to estimate the systematic error due to imperfect clock synchronization (the time offsets are

smoothed, thus the random variation is not usually represented in the results).

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

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

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

7.5. Administrative items

7.5.1. Status

Current

7.5.2. Requestor

This REFC number

7.5.3. Revision

1.0

7.5.4. Revision Date

YYYY-MM-DD

7.6. Comments and Remarks

Additional (Informational) details for this entry

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

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

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

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

8.1. Summary

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

8.1.1. ID (Identifier)

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

8.1.2. Name

OWDelay_Active_IP-UDP-Periodic20m-
Payload142B_RFCXXXXsecY_Seconds_<statistic>

where <statistic> is one of:

- 95Percentile
- Mean
- Min
- Max
- StdDev

OWLoss_Active_IP-UDP-Periodic-
Payload142B_RFCXXXXsecY_Percent_LossRatio

8.1.3. URIs

URL: http:\\www.iana.org\ ... <name>

8.1.4. Description

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

where <statistic> is one of:

- 95Percentile
- Mean
- Min
- Max
- StdDev

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

8.2. Metric Definition

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

8.2.1. Reference Definition

For Delay:

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

[RFC7679]

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

[RFC6049]

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

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

For loss:

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

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

8.2.2. Fixed Parameters

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL: set to 255
  - Protocol: Set to 17 (UDP)
- IPv6 header values:
  - DSCP: set to 0
  - Hop Count: set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum MUST be calculated and included in the header

- o UDP Payload: TWAMP Test Packet Formats, Section 4.1.2 of [RFC5357]
  - * Security features in use influence the number of Padding octets.
  - * 142 octets total, including the TWAMP format (if used)

Other measurement parameters:

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

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

8.3. Method of Measurement

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

8.3.1. Reference Method

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

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

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

The reference method requires some way to distinguish between different packets in a stream to establish correspondence between

sending times and receiving times for each successfully-arriving packet. Sequence numbers or other send-order identification MUST be retained at the Src or included with each packet to disambiguate packet reordering if it occurs.

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

8.3.2. Packet Stream Generation

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

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

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

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

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

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

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

8.3.3. Traffic Filtering (observation) Details

NA

8.3.4. Sampling Distribution

NA

8.3.5. Run-time Parameters and Data Format

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

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

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

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

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

8.3.6. Roles

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

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

8.4. Output

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

8.4.1. Type

<insert name of the output type, raw or a selected summary statistic>

See subsection titles in Reference Definition for Latency Types.

8.4.2. Reference Definition

For all output types ---

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

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

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

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

8.4.2.1. Percentile95

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

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

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

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

95Percentile The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction

digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

8.4.2.2. Mean

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

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

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

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

8.4.2.3. Min

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

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

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

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

8.4.2.4. Max

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

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

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

```
Max = (FiniteDelay [j])

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

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

8.4.2.5. Std_Dev

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

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

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

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

8.4.3. Metric Units

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

- 95Percentile
- Mean

- Min
- Max
- StdDev

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

8.4.4. Calibration

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

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

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

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

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

8.5. Administrative items

8.5.1. Status

Current

8.5.2. Requestor

This RFC number

8.5.3. Revision

1.0

8.5.4. Revision Date

YYYY-MM-DD

8.6. Comments and Remarks

9. ICMP Round-trip Latency and Loss Registry Entries

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

This section specifies four Registry entries with many common columns.

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

9.1. Summary

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

9.1.1. ID (Identifier)

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

9.1.2. Name

RTDelay_Active_IP-ICMP-SendOnRcv_RFCXXXXsecY_Seconds_<statistic>

where <statistic> is one of:

- Mean
- Min
- Max

RTLoss_Active_IP-ICMP-SendOnRcv_RFCXXXXsecY_Percent_LossRatio

9.1.3. URIs

URL: http://<TBD by IANA>/<name>

9.1.4. Description

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

- Mean
- Min
- Max

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

9.1.5. Change Controller

IETF

9.1.6. Version (of Registry Format)

1.0

9.2. Metric Definition

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

9.2.1. Reference Definition

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

[RFC2681]

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

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

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

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

[RFC6673]

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

9.2.2. Fixed Parameters

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

- IPv4 header values:
  - DSCP: set to 0
  - TTL: set to 255
  - Protocol: Set to 01 (ICMP)
- IPv6 header values:

* DSCP: set to 0
* Hop Limit: set to 255
* Protocol: Set to 01 (ICMP)

o ICMP header values:

* Type: 8 (Echo Request)
* Code: 0
* Checksum: the checksum MUST be calculated and included in the header
* (Identifier and Sequence Number set at Run-Time)

o ICMP Payload

* total of 32 bytes of random info

Other measurement parameters:

o Tmax: a loss threshold waiting time

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

9.3. Method of Measurement

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

9.3.1. Reference Method

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

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

packet lost. Lost packets SHALL be designated as having undefined delay, and counted for the RTLoss metric.

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

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

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

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

9.3.2. Packet Stream Generation

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

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

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

dT the duration of the interval for allowed sample start times

T0 the actual start time of the periodic stream

The incT and T0 stream parameters will be specified as Run-time parameters, dT is not used in SendOnRcv.

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

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

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

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

9.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

NA

9.3.4. Sampling Distribution

NA

9.3.5. Run-time Parameters and Data Format

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

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

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

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

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

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

9.3.6. Roles

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

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

9.4. Output

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

9.4.1. Type

See subsection titles in Reference Definition for Latency Types.

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

9.4.2. Reference Definition

For all output types ---

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

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

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

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

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

9.4.2.1. Mean

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

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

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

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

9.4.2.2. Min

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

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

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

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

9.4.2.3. Max

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

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

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

```
Max = (FiniteDelay [j])

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

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

9.4.3. Metric Units

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

- Mean
- Min
- Max

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

9.4.4. Calibration

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

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

Both internal loopback calibration and clock synchronization can be used to estimate the *available accuracy* of the Output Metric Units. For example, repeated loopback delay measurements will reveal the

portion of the Output result resolution which is the result of system noise, and thus inaccurate.

9.5. Administrative items

9.5.1. Status

Current

9.5.2. Requestor

This RFC number

9.5.3. Revision

1.0

9.5.4. Revision Date

YYYY-MM-DD

9.6. Comments and Remarks

None

10. TCP Round-Trip Delay and Loss Registry Entries

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

This section specifies four Registry entries with many common columns.

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

10.1. Summary

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

10.1.1. ID (Identifier)

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

10.1.2. Name

RTDelay_Passive_IP-TCP_RFCXXXXsecY_Seconds_<statistic>

where <statistic> is one of:

- Mean
- Min
- Max

RTDelay_Passive_IP-TCP-HS_RFCXXXXsecY_Seconds_Singleton

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

RTLoss_Passive_IP-TCP_RFCXXXXsecY_Packet_Count

10.1.3. URIs

URL: http://<TBD by IANA>/<name>

10.1.4. Description

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

- Mean
- Min
- Max

RTLoss: This metric assesses the estimated loss count for TCP packets constituting a single connection, exchanged between two hosts. We consider the measurement of round-trip delay based on a single

Observation Point [RFC7011] somewhere in the network. The Output is the estimated Loss Count for the measurement interval.

10.1.5. Change Controller

IETF

10.1.6. Version (of Registry Format)

1.0

10.2. Metric Definition

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

10.2.1. Reference Definitions

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

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

[RFC2681]

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

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

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

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

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

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

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

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

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

RTDelay = RTD_fwd + RTD_rev

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

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

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

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

RTDelay_HS = RTD_HS_fwd + RTD_HS_rev

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

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

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

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

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

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

RTLoss = RTL_fwd + RTL_rev

10.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Traffic Filters:

- IPv4 header values:
  - DSCP: set to 0
  - Protocol: Set to 06 (TCP)
- IPv6 header values:
  - DSCP: set to 0

  * Protocol: Set to 06 (TCP)

- TCP header values:

  * Flags: ACK, SYN, FIN, @@@@@ others??

  * Timestamp Option (TSopt): Set

    + Kind: 8

    + Length: 10 bytes

-

10.3. Method of Measurement

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

10.3.1. Reference Methods

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

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

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

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

of the ACK from B to A, and the corresponding packet with Timestamp echo (TSecr) is the RTD_rev.

Delay Measurement Filtering Heuristics:

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

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

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

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

Method for Inferring Loss:

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

Loss Measurement Filtering Heuristics:

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

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

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

Sources of Error:

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

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

10.3.2. Packet Stream Generation

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

NA

10.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

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

10.3.4. Sampling Distribution

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

10.3.5. Run-time Parameters and Data Format

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

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

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

T0 a time, the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3

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

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

TTL or Hop Limit Set at desired value.

10.3.6. Roles

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

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

10.4. Output

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

10.4.1. Type

See subsection titles in Reference Definition for RTDelay Types.

For RTLoss -- the count of lost packets.

10.4.2. Reference Definition

For all output types ---

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

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

packets exchanged between host A and B effectively ends the interval.

... ...

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

For RTLoss -- the count of lost packets.

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

10.4.2.1. Mean

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

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

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

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

10.4.2.2. Min

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

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

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

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

10.4.2.3. Max

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

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

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

```
Max = (FiniteDelay [j])

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

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

10.4.3. Metric Units

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

- Mean
- Min
- Max

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

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

10.4.4. Calibration

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

10.5. Administrative items

10.5.1. Status

Current

10.5.2. Requestor

This RFC

10.5.3. Revision

1.0

10.5.4. Revision Date

YYYY-MM-DD

10.6. Comments and Remarks

None.

11. Security Considerations

These registry entries represent no known implications for Internet Security. Each referenced Metric contains a Security Considerations section.

12. IANA Considerations

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

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

13. Acknowledgements

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

The authors also acknowledge the constructive reviews and helpful suggestions from Barbara Stark, Juergen Schoenwaelder, Tim Carey, Yaakov Stein, and participants in the LMAP working group. Thanks to Michelle Cotton for her early IANA review, and to Amanda Barber for answering questions related to the presentation of the registry and accessibility of the complete template via URL.

14. References

14.1. Normative References

[I-D.ietf-ippm-metric-registry]
Bagnulo, M., Claise, B., Eardley, P., and A. Morton, "Registry for Performance Metrics", Internet Draft (work in progress) draft-ietf-ippm-metric-registry, 2014.

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

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

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

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

[RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, DOI 10.17487/RFC2680, September 1999, <https://www.rfc-editor.org/info/rfc2680>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

14.2. Informative References

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

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <https://www.rfc-editor.org/info/rfc3611>.

[RFC4148] Stephan, E., "IP Performance Metrics (IPPM) Metrics Registry", BCP 108, RFC 4148, DOI 10.17487/RFC4148, August 2005, <https://www.rfc-editor.org/info/rfc4148>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC 5472, DOI 10.17487/RFC5472, March 2009, <https://www.rfc-editor.org/info/rfc5472>.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, DOI 10.17487/RFC5477, March 2009, <https://www.rfc-editor.org/info/rfc5477>.

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

[RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, DOI 10.17487/RFC6248, April 2011, <https://www.rfc-editor.org/info/rfc6248>.

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

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

[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, DOI 10.17487/RFC6776, October 2012, <https://www.rfc-editor.org/info/rfc6776>.

[RFC6792] Wu, Q., Ed., Hunt, G., and P. Arden, "Guidelines for Use of the RTP Monitoring Framework", RFC 6792, DOI 10.17487/RFC6792, November 2012, <https://www.rfc-editor.org/info/rfc6792>.

[RFC7003] Clark, A., Huang, R., and Q. Wu, Ed., "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting", RFC 7003, DOI 10.17487/RFC7003, September 2013, <https://www.rfc-editor.org/info/rfc7003>.

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

[Strowes] Strowes, S., "Passively Measuring TCP Round Trip Times, Communications of the ACM, Vol. 56 No. 10, Pages 57-64", September 2013.

[Trammell-14]
Trammell, B., "Inline Data Integrity Signals for Passive Measurement, TMA 2014 https://trammell.ch/pdf/qof-tma14.pdf", March 2014.

Authors' Addresses

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

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

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich
ENGLAND

Email: philip.eardley@bt.com

Kevin D'Souza
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 xxxx
Email: kld@att.com

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: September 29, 2019

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

March 28, 2019

Registry for Performance Metrics
draft-ietf-ippm-metric-registry-19

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 29, 2019.

Copyright Notice

Table of Contents

1. 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) concluded WG specified an IANA process for new Information Elements. Some Performance Metrics related Information Elements are proposed on regular basis.

The "Performance Metrics for Other Layers" (PMOL) a concluded WG, defined some Performance Metrics related to Session Initiation Protocol (SIP) voice quality [RFC6035].

It is expected that more Performance Metrics will be defined in the future, not only IP-based metrics, but also metrics which are protocol-specific and application-specific.

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 a new Performance Metric that is very similar, but not quite inter-operable. 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 requests that IANA create and maintain a Performance Metrics Registry, according to the maintenance procedures and the Performance Metrics Registry format defined in this memo. The resulting Performance Metrics Registry is for use by the IETF and others. Although the Registry formatting specifications herein are primarily for registry creation by IANA, any other organization that wishes to create a Performance Metrics Registry MAY use the same formatting specifications for their purposes. The authors make no guarantee of the registry format's applicability to any possible set of Performance Metrics envisaged by other organizations, but encourage others to apply it. In the remainder of this document, unless we explicitly say otherwise, we will refer to the IANA-maintained Performance Metrics Registry as simply the Performance Metrics Registry.

2. Terminology

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

Performance Metric: A Performance Metric is a quantitative measure of performance, targeted to an IETF-specified protocol or targeted to an application transported over an IETF-specified protocol. Examples of Performance Metrics are the FTP response time for a

complete file download, the DNS response time to resolve the IP address, a database logging time, etc. This definition is consistent with the definition of metric in [RFC2330] and broader than the definition of performance metric in [RFC6390].

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

Performance Metrics Registry: The IANA registry containing Registered Performance Metrics.

Proprietary Registry: A set of metrics that are registered in a proprietary registry, as opposed to Performance Metrics Registry.

Performance Metrics Experts: The Performance Metrics Experts is a group of designated experts [RFC8126] selected by the IESG to validate the Performance Metrics before updating the Performance Metrics Registry. The Performance Metrics Experts work closely with IANA.

Parameter: An input factor defined as a variable in the definition of a Performance 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. There are two types of Parameters, Fixed and Run-time parameters. For the Fixed Parameters, the value of the variable is specified in the Performance Metrics Registry entry and different Fixed Parameter values results in different Registered Performance Metrics. For the Run-time Parameters, the value of the variable is defined when the metric measurement method is executed and a given Registered Performance Metric supports multiple values for the parameter. Although Run-time Parameters do not change the fundamental nature of the Performance Metric's definition, some have substantial influence on the network property being assessed and interpretation of the results.

Note: Consider the case of packet loss in the following two Active Measurement Method cases. The first case is packet loss as background loss where the Run-time Parameter set includes a very sparse Poisson stream, and only characterizes the times when packets were lost. Actual user streams likely see much higher loss at these times, due to tail drop or radio errors. The second case is packet loss as inverse of throughput where the Run-time Parameter set includes a very dense, bursty

stream, and characterizes the loss experienced by a stream that approximates a user stream. These are both "loss metrics", but the difference in interpretation of the results is highly dependent on the Run-time Parameters (at least), to the extreme where we are actually using loss to infer its compliment: delivered throughput.

Active Measurement Method: Methods of Measurement conducted on traffic which serves only the purpose of measurement and is generated for that reason alone, and whose traffic characteristics are known a priori. The complete definition of Active Methods is specified in section 3.4 of[RFC7799]. Examples of Active Measurement Methods are the measurement methods for the One way delay metric defined in [RFC7679] and the one for round trip delay defined in [RFC2681].

Passive Measurement Method: Methods of Measurement conducted on network traffic, generated either from the end users or from network elements that would exist regardless whether the measurement was being conducted or not. The complete definition of Passive Methods is specified in section 3.6 of [RFC7799]. One characteristic of Passive Measurement Methods is that sensitive information may be observed, and as a consequence, stored in the measurement system.

Hybrid Measurement Method: Hybrid Methods are Methods of Measurement that use a combination of Active Methods and Passive Methods, to assess Active Metrics, Passive Metrics, or new metrics derived from the a priori knowledge and observations of the stream of interest. The complete definition of Hybrid Methods is specified in section 3.8 of [RFC7799].

3. Scope

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

Metric registry of their own, and may re-use the features of the Performance Metrics Registry defined in this document.

This Performance Metrics Registry is applicable to Performance Metrics issued from Active Measurement, Passive Measurement, and any other form of Performance Metric. This registry is designed to encompass Performance Metrics developed throughout the IETF and especially for the technologies specified in the following working groups: IPPM, XRBLOCK, IPFIX, and BMWG. This document analyzes an prior attempt to set up a Performance Metrics Registry, and the reasons why this design was inadequate [RFC6248]. Finally, this document gives a set of guidelines for requesters and expert reviewers of candidate Registered Performance Metrics.

This document makes no attempt to populate the Performance Metrics Registry with initial entries.

Based on [RFC8126] Section 4.3, this document is processed as Best Current Practice (BCP) [RFC2026].

4. Motivation for a Performance Metrics Registry

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

4.1. 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 Metrics Registry, there are two types of protocols that will use the Performance Metrics in the Performance Metrics Registry during their operation (by referring to the Index values):

- 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 Performance Metrics Registry. One particular example is the LMAP framework [RFC7594]. Using the LMAP terminology, the Performance Metrics Registry is used in the LMAP Control protocol to allow a Controller to request a measurement task to one or more Measurement Agents. In order to enable this use case, the entries of the Performance Metrics 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 requirement heavily constrains the type of entries that are acceptable for the Performance Metrics 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 Metrics Registry, it is possible to properly characterize the measurement result data being reported. Using the LMAP terminology, the Performance Metrics Registry is used in the Report protocol to allow a Measurement Agent to report measurement results to a Collector.

It should be noted that the LMAP framework explicitly allows for using not only the IANA-maintained Performance Metrics Registry but also other registries containing Performance Metrics, either defined by other organizations or private ones. However, others who are creating Registries to be used in the context of an LMAP framework are encouraged to use the Registry format defined in this document, because this makes it easier for developers of LMAP Measurement Agents (MAs) to programmatically use information found in those other Registries' entries.

4.2. Single point of reference for Performance Metrics

A Performance Metrics Registry serves as a single point of reference for Performance Metrics defined in different working groups in the IETF. As we mentioned earlier, there are several WGs that define Performance Metrics in the IETF and it is hard to keep track of all them. This results in multiple definitions of similar Performance Metrics that attempt to measure the same phenomena but in slightly different (and incompatible) ways. Having a registry would allow 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 Performance Metrics, where different entities that request measurements, execute measurements, and report the results can benefit from a common understanding of the referenced Performance Metric.

4.3. Side benefits

There are a couple of side benefits of having such a registry. First, the Performance Metrics Registry could serve as an inventory of useful and used Performance Metrics, that are normally supported by different implementations of measurement agents. Second, the results of measurements using the Performance Metrics would be comparable even if they are performed by different implementations and in different networks, as the Performance Metric is properly defined. BCP 176 [RFC6576] examines whether the results produced by independent implementations are equivalent in the context of evaluating the completeness and clarity of metric specifications. This BCP defines the standards track advancement testing for (active)

IPPM metrics, and the same process will likely suffice to determine whether Registered Performance Metrics are sufficiently well specified to result in comparable (or equivalent) results. Registered Performance Metrics which have undergone such testing SHOULD be noted, with a reference to the test results.

5. Criteria for Performance Metrics Registration

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

1. interpretable by the user.

2. implementable by the software designer,

3. deployable by network operators,

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

6. Performance Metric Registry: Prior attempt

There was a previous attempt to define a metric registry RFC 4148 [RFC4148]. However, it was obsoleted by RFC 6248 [RFC6248] because it was "found to be insufficiently detailed to uniquely identify IPPM metrics... [there was too much] variability possible when characterizing a metric exactly" which led to the RFC4148 registry having "very few users, if any".

A couple of interesting additional quotes from RFC 6248 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 Registered Performance Metric with only a few variable (Run-time) Parameters to be specified by the measurement designer, if any. The idea is that entries in the Performance Metrics Registry stem from different measurement methods which require input (Run-time) parameters to set factors like source and destination addresses (which do not change the fundamental nature of the measurement). The downside of this approach is that it could result in a large number of entries in the Performance Metrics Registry. There is agreement that less is more in this context - it is better to have a reduced set of useful metrics rather than a large set of metrics, some with with questionable usefulness.

6.1. Why this Attempt Will Succeed

As mentioned in the previous section, one of the main issues with the previous registry was that the metrics contained in the registry were too generic to be useful. This document specifies stricter criteria for performance metric registration (see section 6), and imposes a group of Performance Metrics Experts that will provide guidelines to assess if a Performance Metric is properly specified.

Another key difference between this attempt and the previous one is that in this case there is at least one clear user for the Performance Metrics Registry: the LMAP framework and protocol. Because the LMAP protocol will use the Performance Metrics Registry values in its operation, this actually helps to determine if a metric is properly defined. In particular, since we expect that the LMAP control protocol will enable a controller to request a measurement agent to perform a measurement using a given metric by embedding the Performance Metrics Registry 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.

7. Definition of the Performance Metric Registry

This Performance Metrics Registry is applicable to Performance Metrics used for Active Measurement, Passive Measurement, and any other form of Performance Metric. Each category of measurement has unique properties, so some of the columns defined below are not applicable for a given metric category. In this case, the column(s) SHOULD be populated with the "NA" value (Non Applicable). However, the "NA" value MUST NOT be used by any metric in the following columns: Identifier, Name, URI, Status, Requester, Revision, Revision Date, Description. In the future, a new category of metrics could require additional columns, and adding new columns is a recognized form of registry extension. The specification defining the new column(s) MUST give guidelines to populate the new column(s) for existing entries (in general).

The columns of the Performance Metrics Registry are defined below. The columns are grouped into "Categories" to facilitate the use of the registry. Categories are described at the 7.x heading level, and columns are at the 7.x.y heading level. The Figure below illustrates this organization. An entry (row) therefore gives a complete description of a Registered Performance Metric.

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

Registry Categories and Columns, shown as

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

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

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

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

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

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

7.1. Summary Category

7.1.1. Identifier

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

The Registered Performance Metric unique identifier is an unbounded integer (range 0 to infinity).

The Identifier 0 should be Reserved. The Identifier values from 64512 to 65536 are reserved for private use.

When adding newly Registered Performance Metrics to the Performance Metrics Registry, IANA SHOULD assign the lowest available identifier to the new Registered Performance Metric.

If a Performance Metrics Expert providing review determines that there is a reason to assign a specific numeric identifier, possibly leaving a temporary gap in the numbering, then the Performance Expert SHALL inform IANA of this decision.

7.1.2. Name

As the name of a Registered Performance Metric is the first thing a potential human implementor will use when determining whether it is suitable for their measurement study, it is important to be as precise and descriptive as possible. In future, users will review the names to determine if the metric they want to measure has already been registered, or if a similar entry is available as a basis for creating a new entry.

Names are composed of the following elements, separated by an underscore character "_":

MetricType_Method_SubTypeMethod_... Spec_Units_Output

- MetricType: a combination of the directional properties and the metric measured, such as:

  RTDelay (Round Trip Delay)

  RTDNS (Response Time Domain Name Service)

  RLDNS (Response Loss Domain Name Service)

  OWDelay (One Way Delay)

  RTLoss (Round Trip Loss)

  OWLoss (One Way Loss)

  OWPDV (One Way Packet Delay Variation)

  OWIPDV (One Way Inter-Packet Delay Variation)

  OWReorder (One Way Packet Reordering)

  OWDuplic (One Way Packet Duplication)

  OWBTC (One Way Bulk Transport Capacity)

  OWMBM (One Way Model Based Metric)

  SPMonitor (Single Point Monitor)

MPMonitor (Multi-Point Monitor)

- Method: One of the methods defined in [RFC7799], such as:

  Active (depends on a dedicated measurement packet stream and observations of the stream)

  Passive (depends *solely* on observation of one or more existing packet streams)

  HybridType1 (obervations on one stream that combine both active and passive methods)

  HybridType2 (obervations on two or more streams that combine both active and passive methods)

  Spatial (Spatial Metric of RFC5644)

- SubTypeMethod: One or more sub-types to further describe the features of the entry, such as:

  ICMP (Internet Control Message Protocol)

  IP (Internet Protocol)

  DSCPxx (where xx is replaced by a Diffserv code point)

  UDP (User Datagram Protocol)

  TCP (Transport Control Protocol)

  QUIC (QUIC transport protocol)

  HS (Hand-Shake, such as TCP's 3-way HS)

  Poisson (Packet generation using Poisson distribution)

  Periodic (Periodic packet generation)

  SendOnRcv (Sender keeps one packet in-transit by sending when previous packet arrives)

  PayloadxxxxB (where xxxx is replaced by an integer, the number of octets in the Payload))

  SustainedBurst (Capacity test, worst case)

  StandingQueue (test of bottleneck queue behavior)

SubTypeMethod values are separated by a hyphen "-" character, which indicates that they belong to this element, and that their order is unimportant when considering name uniqueness.

- Spec: RFC number and major section number that specifies this Registry entry in the form RFCXXXXsecY, such as RFC7799sec3. Note: the RFC number is not the Primary Reference specification for the metric definition, such as [RFC7679] for One-way Delay; it will contain the placeholder "RFCXXXXsecY" until the RFC number is assigned to the specifying document, and would remain blank in private registry entries without a corresponding RFC.

- Units: The units of measurement for the output, such as:

  Seconds

  Ratio (unitless)

  Percent (value multiplied by 100)

  Logical (1 or 0)

  Packets

  BPS (Bits per Second)

  PPS (Packets per Second)

  EventTotal (for unit-less counts)

  Multiple (more than one type of unit)

  Enumerated (a list of outcomes)

  Unitless

- Output: The type of output resulting from measurement, such as:

  Singleton

  Raw (multiple Singletons)

  Count

  Minimum

  Maximum

Median

Mean

95Percentile (95th Percentile)

99Percentile (99th Percentile)

StdDev (Standard Deviation)

Variance

PFI (Pass, Fail, Inconclusive)

FlowRecords (descriptions of flows observed)

LossRatio (lost packets to total packets, <=1)

An example is:

RTDelay_Active_IP-UDP-Periodic_RFCXXXXsecY_Seconds_95Percentile

as described in section 4 of [I-D.ietf-ippm-initial-registry].

Note that private registries following the format described here SHOULD use the prefix "Priv_" on any name to avoid unintended conflicts (further considerations are described in section 10). Private registry entries usually have no specifying RFC, thus the Spec: element has no clear interpretation.

7.1.3. URIs

The URIs column MUST contain a URL [RFC3986] and uniquely identifies and locates the metric entry so it is accessible through the Internet. The URL points to a file containing all the human-readable information for one registry entry. The URL SHALL reference a target file that is HTML-formated and contains URLs to referenced sections of HTML-ized RFCs. These target files for different entries can be more easily edited and re-used when preparing new entries. The exact form of the URL for each target file will be determined by IANA and reside on "iana.org". The major sections of [I-D.ietf-ippm-initial-registry] provide an example of a target file in HTML form (sections 4 and higher).

7.1.4. Description

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

7.1.5. Reference

This entry gives the specification containing the candidate registry entry which was reviewed and agreed, if such an RFC or other specification exists.

7.1.6. Change Controller

This entry names the entity responsible for approving revisions to the registry entry, and SHALL provide contact information (for an individual, where appropriate).

7.1.7. Version (of Registry Format)

This entry gives the version number for the registry format used. Formats complying with this memo MUST use 1.0. The version number SHALL not change unless a new RFC is published that changes the registry format.

7.2. 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.

7.2.1. Reference Definition

This entry provides a reference (or references) to the relevant section(s) of the document(s) that define the metric, as well as any supplemental information needed to ensure an unambiguous definition for implementations. The reference needs to be an immutable document, such as an RFC; for other standards bodies, it is likely to be necessary to reference a specific, dated version of a specification.

7.2.2. Fixed Parameters

Fixed Parameters are Parameters whose value must be specified in the Performance Metrics Registry. The measurement system uses these values.

Where referenced metrics supply a list of Parameters as part of their descriptive template, a sub-set of the Parameters will be designated as Fixed Parameters. As an example for active metrics, Fixed Parameters determine most or all of the IPPM Framework convention "packets of Type-P" as described in [RFC2330], such as transport protocol, payload length, TTL, etc. An example for passive metrics is for RTP packet loss calculation that relies on the validation of a packet as RTP which is a multi-packet validation controlled by MIN_SEQUENTIAL as defined by [RFC3550]. Varying MIN_SEQUENTIAL values can alter the loss report and this value could be set as a Fixed Parameter.

Parameters MUST have well-defined names. For human readers, the hanging indent style is preferred, and any Parameter names and definitions that do not appear in the Reference Method Specification MUST appear in this column (or Run-time Parameters column).

Parameters MUST have a well-specified data format.

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

7.3. Method of Measurement Category

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

7.3.1. Reference Method

This entry provides references to relevant sections of 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.

7.3.2. Packet Stream Generation

This column applies to Performance Metrics that generate traffic as part of their Measurement Method, including but not necessarily limited to Active metrics. The generated traffic is referred as a stream and this column describes its characteristics.

Each entry for this column contains the following information:

- Value: The name of the packet stream scheduling discipline
- Reference: the specification where the parameters of the stream are defined

The packet generation stream may require parameters such as the the average packet rate and distribution truncation value for streams with Poisson-distributed inter-packet sending times. In case such parameters are needed, they should be included either in the Fixed parameter column or in the run time parameter column, depending on wether they will be fixed or will be an input for the metric.

The simplest example of stream specification is Singleton scheduling (see [RFC2330]), where a single atomic measurement is conducted. Each atomic measurement could consist of sending a single packet (such as a DNS request) or sending several packets (for example, to request a webpage). Other streams support a series of atomic measurements in a "sample", with a schedule defining the timing between each transmitted packet and subsequent measurement. Principally, two different streams are used in IPPM metrics, Poisson distributed as described in [RFC2330] and Periodic as described in [RFC3432]. Both Poisson and Periodic have their own unique parameters, and the relevant set of parameters names and values should be included either in the Fixed Parameters column or in the Run-time parameter column.

7.3.3. Traffic Filter

This column applies to Performance Metrics that observe packets flowing through (the device with) the measurement agent i.e. that is not necessarily addressed to the measurement agent. This includes but is not limited to Passive Metrics. The filter specifies the traffic that is measured. This includes protocol field values/ ranges, such as address ranges, and flow or session identifiers.

The traffic filter itself depends on needs of the metric itself and a balance of operators measurement needs and user's need for privacy. Mechanics for conveying the filter criteria might be the BPF (Berkley Packet Filter) or PSAMP [RFC5475] Property Match Filtering which

reuses IPFIX [RFC7012]. An example BPF string for matching TCP/80 traffic to remote destination net 192.0.2.0/24 would be "dst net 192.0.2.0/24 and tcp dst port 80". More complex filter engines might be supported by the implementation that might allow for matching using Deep Packet Inspection (DPI) technology.

The traffic filter includes the following information:

Type: the type of traffic filter used, e.g. BPF, PSAMP, OpenFlow rule, etc. as defined by a normative reference

Value: the actual set of rules expressed

7.3.4. Sampling Distribution

The sampling distribution defines out of all the packets that match the traffic filter, which one of those are actually used for the measurement. One possibility is "all" which implies that all packets matching the Traffic filter are considered, but there may be other sampling strategies. It includes the following information:

Value: the name of the sampling distribution

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

The sampling distribution may require parameters. In case such parameters are needed, they should be included either in the Fixed parameter column or in the run time parameter column, depending on whether they will be fixed or will be an input for the metric.

Sampling and Filtering Techniques for IP Packet Selection are documented in the PSAMP (Packet Sampling) [RFC5475], while the Framework for Packet Selection and Reporting, [RFC5474] provides more background information. The sampling distribution parameters might be expressed in terms of the Information Model for Packet Sampling Exports, [RFC5477], and the Flow Selection Techniques, [RFC7014].

7.3.5. Run-time Parameters

Run-Time Parameters are Parameters that must be determined, configured into the measurement system, and reported with the results for the context to be complete. However, the values of these parameters is not specified in the Performance Metrics Registry (like the Fixed Parameters), rather these parameters are listed as an aid to the measurement system implementer or user (they must be left as variables, and supplied on execution).

Where metrics supply a list of Parameters as part of their descriptive template, a sub-set of the Parameters will be designated as Run-Time Parameters.

Parameters MUST have well defined names. For human readers, the hanging indent style is preferred, and the names and definitions that do not appear in the Reference Method Specification MUST appear in this column.

A Data Format for each Run-time Parameter MUST be specified in this column, to simplify the control and implementation of measurement devices. For example, parameters that include an IPv4 address can be encoded as a 32 bit integer (i.e. binary base64 encoded value) or ip-address as defined in [RFC6991]. The actual encoding(s) used must be explicitly defined for each Run-time parameter. IPv6 addresses and options MUST be accomodated, allowing Registered Metrics to be used in either address family.

Examples of Run-time Parameters include IP addresses, measurement point designations, start times and end times for measurement, and other information essential to the method of measurement.

7.3.6. Role

In some methods of measurement, there may be several roles defined, e.g., for a one-way packet delay active measurement there is one measurement agent that generates the packets and another agent that receives the packets. This column contains the name of the Role(s) for this particular entry. In the one-way delay example above, there should be two entries in the Role registry column, one for each Role (Source and Destination). When a measurement agent is instructed to perform the "Source" Role for one-way delay metric, the agent knows that it is required to generate packets. The values for this field are defined in the reference method of measurement (and this frequently results in abbreviated role names such as "Src").

When the Role column of a registry entry defines more than one Role, then the Role SHALL be treated as a Run-time Parameter and supplied for execution. It should be noted that the LMAP framework [RFC7594] distinguishes the Role from other Run-time Parameters, and defines a special parameter "Roles" inside the registry-grouping function list in the LMAP YANG model[RFC8194].

7.4. Output Category

For entries which involve a stream and many singleton measurements, a statistic may be specified in this column to summarize the results to

a single value. If the complete set of measured singletons is output, this will be specified here.

Some metrics embed one specific statistic in the reference metric definition, while others allow several output types or statistics.

7.4.1. Type

This column contains the name of the output type. The output type defines a single type of result that the metric produces. It can be the raw results (packet send times and singleton metrics), or it can be a summary statistic. The specification of the output type MUST define the format of the output. In some systems, format specifications will simplify both measurement implementation and collection/storage tasks. Note that if two different statistics are required from a single measurement (for example, both "Xth percentile mean" and "Raw"), then a new output type must be defined ("Xth percentile mean AND Raw"). See the Naming section above for a list of Output Types.

7.4.2. Reference Definition

This column contains a pointer to the specification(s) where the output type and format are defined.

7.4.3. Metric Units

The measured results must be expressed using some standard dimension or units of measure. This column provides the units.

When a sample of singletons (see Section 11 of[RFC2330] for definitions of these terms) is collected, this entry will specify the units for each measured value.

7.4.4. Calibration

Some specifications for Methods of Measurement include the possibility to perform an error calibration. Section 3.7.3 of [RFC7679] is one example. In the registry entry, this field will identify a method of calibration for the metric, and when available, the measurement system SHOULD perform the calibration when requested and produce the output with an indication that it is the result of a calbration method. In-situ calibration could be enabled with an internal loopback that includes as much of the measurement system as possible, performs address manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

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

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

7.5. Administrative information

7.5.1. Status

The status of the specification of this Registered Performance Metric. Allowed values are 'current' and 'deprecated'. All newly defined Information Elements have 'current' status.

7.5.2. Requester

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

7.5.3. Revision

The revision number of a Registered Performance Metric, starting at 0 for Registered Performance Metrics at time of definition and incremented by one for each revision.

7.5.4. Revision Date

The date of acceptance or the most recent revision for the Registered Performance Metric. The date SHALL be determined by the reviewing Performance Metrics Expert in the case of Expert Review, or by IANA in the case of Standards Action.

7.6. Comments and Remarks

Besides providing additional details which do not appear in other categories, this open Category (single column) allows for unforeseen issues to be addressed by simply updating this informational entry.

8. The Life-Cycle of Registered Performance Metrics

Once a Performance Metric or set of Performance Metrics has been identified for a given application, candidate Performance Metrics Registry entry specifications prepared in accordance with Section 7 should be submitted to IANA to follow the process for review by the Performance Metric Experts, as defined below. This process is also used for other changes to the Performance Metrics Registry, such as deprecation or revision, as described later in this section.

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

8.1. Adding new Performance Metrics to the Performance Metrics Registry

Requests to add Registered Performance Metrics in the Performance Metrics Registry SHALL be submitted to IANA, which forwards the request to a designated group of experts (Performance Metric Experts) appointed by the IESG; these are the reviewers called for by the Expert Review [RFC8126]policy defined for the Performance Metrics Registry. The Performance Metric Experts review the request for such things as compliance with this document, compliance with other applicable Performance Metric-related RFCs, and consistency with the currently defined set of Registered Performance Metrics.

Submission to IANA MAY be the result of IETF Standards Action, where an approved Internet Draft proposes one or more Registered Performance Metrics to be added to the Performance Metrics Registry, including the text of the proposed Registered Performance Metric(s).

Authors of proposed Registered Performance Metrics SHOULD review compliance with the specifications in this document to check their submissions before sending them to IANA.

At least one Performance Metric Expert should endeavor to complete referred reviews in a timely manner. If the request is acceptable, the Performance Metric Experts signify their approval to IANA, and IANA updates the Performance Metrics Registry. If the request is not acceptable, the Performance Metric Experts MAY coordinate with the requester to change the request to be compliant, otherwise IANA SHALL coordinate resolution of issues on behalf of the expert. The Performance Metric Experts MAY choose to reject clearly frivolous or inappropriate change requests outright, but such exceptional circumstances should be rare.

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

Decisions by the Performance Metric Experts may be appealed as in Section 7 of [RFC8126].

8.2. Revising Registered Performance Metrics

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

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

In addition, no policy is defined for revising the Performance Metric entries in the IANA Regsirty or addressing errors therein. To be clear, changes and deprecations within the Performance Metrics Registry are not encouraged, and should be avoided to the extent possible. However, in recognition that change is inevitable, the provisions of this section address the need for revisions.

Revisions are initiated by sending a candidate Registered Performance Metric definition to IANA, as in Section 8.1, identifying the existing Performance Metrics Registry entry, and explaining how and why the existing entry shuold be revised.

The primary requirement in the definition of procedures for managing changes to existing Registered Performance Metrics is avoidance of measurement interoperability problems; the Performance Metric Experts must work to maintain interoperability above all else. Changes to Registered Performance Metrics may only be done in an interoperable way; necessary changes that cannot be done in a way to allow interoperability with unchanged implementations MUST result in the creation of a new Registered Performance Metric (with a new Name, replacing the RFCXXXXsecY portion of the name) and possibly the deprecation of the earlier metric.

A change to a Registered Performance Metric SHALL be determined to be backward-compatible 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 a Performance Metric revision is deemed permissible and backward-compatible by the Performance Metric Experts, according to the rules in this document, IANA SHOULD execute the change(s) in the Performance Metrics Registry. The requester of the change is appended to the original requester in the Performance Metrics Registry. The Name of the revised Registered Performance Metric, including the RFCXXXXsecY portion of the name, SHALL remain unchamged (even when the change is the result of IETF Standards Action; the revised registry entry SHOULD reference the new RFC in an appropriate category and column).

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

When a revised Registered Performance Metric is accepted into the Performance Metrics Registry, the date of acceptance of the most recent revision is placed into the revision Date column of the registry for that Registered Performance Metric.

Where applicable, additions to Registered Performance Metrics in the form of text Comments or Remarks should include the date, but such additions may not constitute a revision according to this process.

Older version(s) of the updated metric entries are kept in the registry for archival purposes. The older entries are kept with all fields unmodified (version, revision date) except for the status field that SHALL be changed to "Deprecated".

8.3. Deprecating Registered Performance Metrics

Changes that are not permissible by the above criteria for Registered Performance Metric's revision may only be handled by deprecation. A Registered Performance Metric MAY be deprecated and replaced when:

1. the Registered Performance Metric definition has an error or shortcoming that cannot be permissibly changed as in Section 8.2 Revising Registered Performance Metrics; or

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

A request for deprecation is sent to IANA, which passes it to the Performance Metric Experts for review. When deprecating an Performance Metric, the Performance Metric description in the Performance Metrics Registry must be updated to explain the deprecation, as well as to refer to any new Performance Metrics created to replace the deprecated Performance Metric.

The revision number of a Registered Performance Metric is incremented upon deprecation, and the revision Date updated, as with any revision.

The use of deprecated Registered Performance Metrics should result in a log entry or human-readable warning by the respective application.

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

The deprecated entries are kept with all fields unmodified, except the version, revision date, and the status field (changed to "Deprecated").

9. Security considerations

This draft defines a registry structure, and does not itself introduce any new security considerations for the Internet. The definition of Performance Metrics for this registry may introduce some security concerns, but the mandatory references should have their own considerations for secuity, and such definitions should be reviewed with security in mind if the security considerations are not covered by one or more reference standards.

10. IANA Considerations

With the background and processes described in earlier sections, this document requests the following IANA Actions. Note that mock-ups of the implementation of this set of requests have been prepared with IANA's help during development of this memo, and have been captured in the Proceedings of IPPM working group sessions.

10.1. Registry Group

The new registry group SHALL be named, "PERFORMANCE METRICS Group".

10.2. Performance Metric Name Elements

This document specifies the procedure for Performance Metrics Name Element Registry setup. IANA is requested to create a new set of registries for Performance Metric Name Elements called "Registered Performance Metric Name Elements". Each Registry, whose names are listed below:

MetricType:

Method:

SubTypeMethod:

Spec:

Units:

Output:

will contain the current set of possibilities for Performance Metrics Registry Entry Names.

To populate the Registered Performance Metric Name Elements at creation, the IANA is asked to use the lists of values for each name element listed in Section 7.1.2. The Name Elements in each registry are case-sensitive.

When preparing a Metric entry for Registration, the developer SHOULD choose Name elements from among the registered elements. However, if the proposed metric is unique in a significant way, it may be necessary to propose a new Name element to properly describe the metric, as described below.

A candidate Metric Entry RFC or document for Expert Review would propose one or more new element values required to describe the

unique entry, and the new name element(s) would be reviewed along with the metric entry. New assignments for Registered Performance Metric Name Elements will be administered by IANA through Expert Review [RFC8126], i.e., review by one of a group of experts, the Performance Metric Experts, who are appointed by the IESG upon recommendation of the Transport Area Directors.

10.3. New Performance Metrics Registry

This document specifies the procedure for Performance Metrics Registry setup. IANA is requested to create a new registry for Performance Metrics called "Performance Metrics Registry". This Registry will contain the following Summary columns:

Identifier:

Name:

URIs:

Description:

Reference:

Change Controller:

Version:

Descriptions of these columns and additional information found in the template for registry entries (categories and columns) are further defined in section Section 7.

The "Identifier" 0 should be Reserved. "The Identifier" values from 64512 to 65536 are reserved for private use.

Names starting with the prefix Priv_ are reserved for private use, and are not considered for registration. The "Name" column entries are further defined in section Section 7.

The "URIs" column will have a URL to the full template of each registry entry. The Registry Entry text SHALL be HTML-ized to aid the reader, with links to reference RFCs (similar to the way that Internet Drafts are HTML-ized, the same tool can perform the function).

The "Reference" column will include an RFC number, an approved specification designator from another standards body, or the contact person.

New assignments for Performance Metrics Registry will be administered by IANA through Expert Review [RFC8126], i.e., review by one of a group of experts, the Performance Metric Experts, who are appointed by the IESG upon recommendation of the Transport Area Directors, or by Standards Action. The experts can be initially drawn from the Working Group Chairs, document editors, and members of the Performance Metrics Directorate, among other sources of experts.

Extensions of the Performance Metrics Registry require IETF Standards Action. Only one form of registry extension is envisaged:

1. Adding columns, or both categories and columns, to accommodate unanticipated aspects of new measurements and metric categories.

If the Performance Metrics Registry is extended in this way, the Version number of future entries complying with the extension SHALL be incremented (either in the unit or tenths digit, depending on the degree of extension.

11. Acknowledgments

Thanks to Brian Trammell and Bill Cerveny, IPPM chairs, for leading some brainstorming sessions on this topic. Thanks to Barbara Stark and Juergen Schoenwaelder for the detailed feedback and suggestions. Thanks to Andrew McGregor for suggestions on metric naming. Thanks to Michelle Cotton for her early IANA review, and to Amanda Barber for answering questions related to the presentation of the registry and accessibility of the complete template via URL.

12. References

12.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

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

[RFC2141] Moats, R., "URN Syntax", RFC 2141, DOI 10.17487/RFC2141, May 1997, <https://www.rfc-editor.org/info/rfc2141>.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4148] Stephan, E., "IP Performance Metrics (IPPM) Metrics Registry", BCP 108, RFC 4148, DOI 10.17487/RFC4148, August 2005, <https://www.rfc-editor.org/info/rfc4148>.

[RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, DOI 10.17487/RFC6248, April 2011, <https://www.rfc-editor.org/info/rfc6248>.

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

[RFC6576] Geib, R., Ed., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, DOI 10.17487/RFC6576, March 2012, <https://www.rfc-editor.org/info/rfc6576>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[I-D.ietf-ippm-initial-registry] Morton, A., Bagnulo, M., Eardley, P., and K. D'Souza, "Initial Performance Metrics Registry Entries", draft-ietf-ippm-initial-registry-10 (work in progress), March 2019.

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

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

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

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

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <https://www.rfc-editor.org/info/rfc3611>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC5474] Duffield, N., Ed., Chiou, D., Claise, B., Greenberg, A., Grossglauser, M., and J. Rexford, "A Framework for Packet Selection and Reporting", RFC 5474, DOI 10.17487/RFC5474, March 2009, <https://www.rfc-editor.org/info/rfc5474>.

[RFC5475] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection", RFC 5475, DOI 10.17487/RFC5475, March 2009, <https://www.rfc-editor.org/info/rfc5475>.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, DOI 10.17487/RFC5477, March 2009, <https://www.rfc-editor.org/info/rfc5477>.

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

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

[RFC6035] Pendleton, A., Clark, A., Johnston, A., and H. Sinnreich, "Session Initiation Protocol Event Package for Voice Quality Reporting", RFC 6035, DOI 10.17487/RFC6035, November 2010, <https://www.rfc-editor.org/info/rfc6035>.

[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, DOI 10.17487/RFC6776, October 2012, <https://www.rfc-editor.org/info/rfc6776>.

[RFC6792] Wu, Q., Ed., Hunt, G., and P. Arden, "Guidelines for Use of the RTP Monitoring Framework", RFC 6792, DOI 10.17487/RFC6792, November 2012, <https://www.rfc-editor.org/info/rfc6792>.

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

[RFC7003] Clark, A., Huang, R., and Q. Wu, Ed., "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting", RFC 7003, DOI 10.17487/RFC7003, September 2013, <https://www.rfc-editor.org/info/rfc7003>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <https://www.rfc-editor.org/info/rfc7012>.

[RFC7014] D'Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow Selection Techniques", RFC 7014, DOI 10.17487/RFC7014, September 2013, <https://www.rfc-editor.org/info/rfc7014>.

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

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

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

[RFC8194] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", RFC 8194, DOI 10.17487/RFC8194, August 2017, <https://www.rfc-editor.org/info/rfc8194>.

Authors' Addresses

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Email: bclaise@cisco.com

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich
ENGLAND

Email: philip.eardley@bt.com

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

Email: acmorton@att.com

Aamer Akhter
Consultant
118 Timber Hitch
Cary, NC
USA

Email: aakhter@gmail.com

IP Performance Working Group
Internet-Draft
Intended status: Experimental
Expires: March 19, 2018

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

September 15, 2017

Model Based Metrics for Bulk Transport Capacity
draft-ietf-ippm-model-based-metrics-13.txt

Abstract

We introduce a new class of Model Based Metrics designed to assess if a complete Internet path can be expected to meet a predefined Target Transport Performance by applying a suite of IP diagnostic tests to successive subpaths. The subpath-at-a-time tests can be robustly applied to critical infrastructure, such as network interconnections or even individual devices, to accurately detect if any part of the infrastructure will prevent paths traversing it from meeting the Target Transport Performance.

Model Based Metrics rely on mathematical models to specify a Targeted Suite of IP Diagnostic tests, designed to assess whether common transport protocols can be expected to meet a predetermined Target Transport Performance over an Internet path.

For Bulk Transport Capacity the IP diagnostics are built using test streams and statistical criteria for evaluating the packet transfer that mimic TCP over the complete path. The temporal structure of the test stream (bursts, etc) mimic TCP or other transport protocol carrying bulk data over a long path. However they are constructed to be independent of the details of the subpath under test, end systems or applications. Likewise the success criteria evaluates the packet transfer statistics of the subpath against criteria determined by protocol performance models applied to the Target Transport Performance of the complete path. The success criteria also does not depend on the details of the subpath, end systems or application.

Status of This Memo

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/.

Copyright Notice

Table of Contents

1. Introduction

Model Based Metrics (MBM) rely on peer-reviewed mathematical models to specify a Targeted Suite of IP Diagnostic tests, designed to assess whether common transport protocols can be expected to meet a predetermined Target Transport Performance over an Internet path. This note describes the modeling framework to derive the test parameters for assessing an Internet path's ability to support a predetermined Bulk Transport Capacity.

Each test in the Targeted IP Diagnostic Suite (TIDS) measures some aspect of IP packet transfer needed to meet the Target Transport Performance. For Bulk Transport Capacity the TIDS includes IP diagnostic tests to verify that there is: sufficient IP capacity (data rate); sufficient queue space at bottlenecks to absorb and deliver typical transport bursts; and that the background packet loss ratio is low enough not to interfere with congestion control; and other properties described below. Unlike typical IPPM metrics which yield measures of network properties, Model Based Metrics nominally yield pass/fail evaluations of the ability of standard transport

protocols to meet the specific performance objective over some network path.

In most cases, the IP diagnostic tests can be implemented by combining existing IPPM metrics with additional controls for generating test streams having a specified temporal structure (bursts or standing queues caused by constant bit rate streams, etc.) and statistical criteria for evaluating packet transfer. The temporal structure of the test streams mimic transport protocol behavior over the complete path; the statistical criteria models the transport protocol's response to less than ideal IP packet transfer. In control theory terms, the tests are "open loop". Note that running a test requires the coordinated activity of sending and receiving measurement points.

This note addresses Bulk Transport Capacity. It describes an alternative to the approach presented in "A Framework for Defining Empirical Bulk Transfer Capacity Metrics" [RFC3148]. Other Model Based Metrics may cover other applications and transports, such as VoIP over UDP and RTP, and new transport protocols.

This note assumes a traditional Reno TCP style self clocked, window controlled transport protocol that uses packet loss and ECN CE marks for congestion feedback. There are currently some experimental protocols and congestion control algorithms that are rate based or otherwise fall outside of these assumptions. In the future these new protocols and algorithms may call for revised models.

The MBM approach, mapping Target Transport Performance to a Targeted IP Diagnostic Suite (TIDS) of IP tests, solves some intrinsic problems with using TCP or other throughput maximizing protocols for measurement. In particular all throughput maximizing protocols (and TCP congestion control in particular) cause some level of congestion in order to detect when they have reached the available capacity limitation of the network. This self inflicted congestion obscures the network properties of interest and introduces non-linear dynamic equilibrium behaviors that make any resulting measurements useless as metrics because they have no predictive value for conditions or paths different than that of the measurement itself. In order to prevent these effects it is necessary to avoid the effects of TCP congestion control in the measurement method. These issues are discussed at length in Section 4. Readers whom are unfamiliar with basic properties of TCP and TCP-like congestion control may find it easier to start at Section 4 or Section 4.1.

A Targeted IP Diagnostic Suite does not have such difficulties. IP diagnostics can be constructed such that they make strong statistical statements about path properties that are independent of the

measurement details, such as vantage and choice of measurement points.

1.1. Version Control

RFC Editor: Please remove this entire subsection prior to publication.

REF Editor: The reference to draft-ietf-tcpm-rack is to attribute an idea. This document should not block waiting for the completion of that one.

Please send comments about this draft to ippm@ietf.org. See http://goo.gl/02tkD for more information including: interim drafts, an up to date todo list and information on contributing.

Formatted: Fri Sep 15 15:07:50 PDT 2017

Changes since -11 draft:

- (From IESG review comments.)
- Ben Campbell: Shorten the Abstract.
- Mirja Kuhlewind: Reduced redundancy. (See message)
- MK: Mention open loop in the introduction.
- MK: Spelled out ECN and reference RFC3168.
- MK: Added a paragraph to the introduction about assuming a traditional self clocked, window controlled transport protocol.
- MK: Added language about initial window to the list at about bursts at the end of section 4.1.
- MK: Network power is defined in the terminology section.
- MK: The introduction mention coordinated activity of both endpoints.
- MK: The security section restates that some of the tests are not intended for frequent monitoring tests as the high load can impact other traffic negatively.
- MK: Restored "Informative References" section name.
- And a few minor nits.

Changes since -10 draft:

- A few more nits from various sources.
- (From IETF LC review comments.)
- David Mandelberg: design metrics to prevent DDOS.
- From Robert Sparks:
  - Remove all legacy 2119 language.
  - Fixed Xr notation inconsistency.
  - Adjusted abstract: tests are only partially specified.

    * Avoid rather than suppress the effects of congestion control
    * Removed the unnecessary, excessively abstract and unclear thought about IP vs TCP measurements.
    * Changed "thwarted" to "not fulfilled".
    * Qualified language about burst models.
    * Replaced "infinitesimal" with other language.
    * Added citations for the reordering strawman.
    * Pointed out that pseudo CBR tests depend on self clock.
    * Fixed some run on sentences.
  o Update language to reflect RFC7567, AQM recommendations.
  o Suggestion from Merry Mou (MIT)

Changes since -09 draft:

  o Five last minute editing nits.

Changes since -08 draft:

  o Language, spelling and usage nits.
  o Expanded the abstract describe the models.
  o Remove superfluous standards like language
  o Remove superfluous "future technology" language.
  o Interconnects -> network interconnections.
  o Added more labels to Figure 1.
  o Defined Bulk Transport.
  o Clarified "implied bottleneck IP capacity"
  o Clarified the history of the BTC metrics.
  o Clarified stochastic vs non-stochastic test traffic generation.
  o Reworked Fig 2 and 6.1 "Mimicking slowstart"
  o Described the unsynchronized parallel stream failure case.
  o Discussed how to measure devices that use virtual queues.
  o Changed section 8.5.2 (Streaming Media) to be Passive Measurements.

Changes since -07 draft:

  o Sharpened the use of "statistical criteria"
  o Sharpened the definition of test_window, and removed related redundant text in several places
  o Clarified "equilibrium" as "dynamic equilibrium, similar to processes observed in chemistry"
  o Properly explained "Heisenberg" as "observer effect"
  o Added the observation from RFC 6576 that HW and SW congestion control implementations do not generally give the same results.
  o Noted that IP and application metrics differ as to how overhead is handled. MBM is explicit about how it handles overhead.
  o Clarified the language and added a new reference about the problems caused by token bucket policers.

- Added an subsection in the example that comments on some of issues that need to be mentioned in a future usage or applicability doc.
- Updated ippm-2680-bis to RFC7680
- Many terminology, punctuation and spelling nits.

Changes since -06 draft:

- More language nits:
  - "Targeted IP Diagnostic Suite (TIDS)" replaces "Targeted Diagnostic Suite (TDS)".
  - "implied bottleneck IP capacity" replaces "implied bottleneck IP rate".
  - Updated to ECN CE Marks.
  - Added "specified temporal structure"
  - "test stream" replaces "test traffic"
  - "packet transfer" replaces "packet delivery"
  - Reworked discussion of slowstart, bursts and pacing.
  - RFC 7567 replaces RFC 2309.

Changes since -05 draft:

- Wordsmithing on sections overhauled in -05 draft.
- Reorganized the document:
  - Relocated subsection "Preconditions".
  - Relocated subsection "New Requirements relative to RFC 2330".
- Addressed nits and not so nits by Ruediger Geib. (Thanks!)
- Substantially tightened the entire definitions section.
- Many terminology changes, to better conform to other docs :
  - IP rate and IP capacity (following RFC 5136) replaces various forms of link data rate.
  - subpath replaces link.
  - target_window_size replaces target_pipe_size.
  - implied bottleneck IP rate replaces effective bottleneck link rate.
  - Packet delivery statistics replaces delivery statistics.

Changes since -04 draft:

- The introduction was heavily overhauled: split into a separate introduction and overview.
- The new shorter introduction:
  - Is a problem statement;
  - This document provides a framework;
  - That it replaces TCP measurement by IP tests;

    * That the results are pass/fail.
  o Added a diagram of the framework to the overview
  o and introduces all of the elements of the framework.
  o Renumbered sections, reducing the depth of some section numbers.
  o Updated definitions to better agree with other documents:

    * Reordered section 2
    * Bulk [data] performance -> Bulk Transport Capacity, everywhere including the title.
    * loss rate and loss probability -> packet loss ratio
    * end-to-end path -> complete path
    * [end-to-end][target] performance -> Target Transport Performance
    * load test -> capacity test

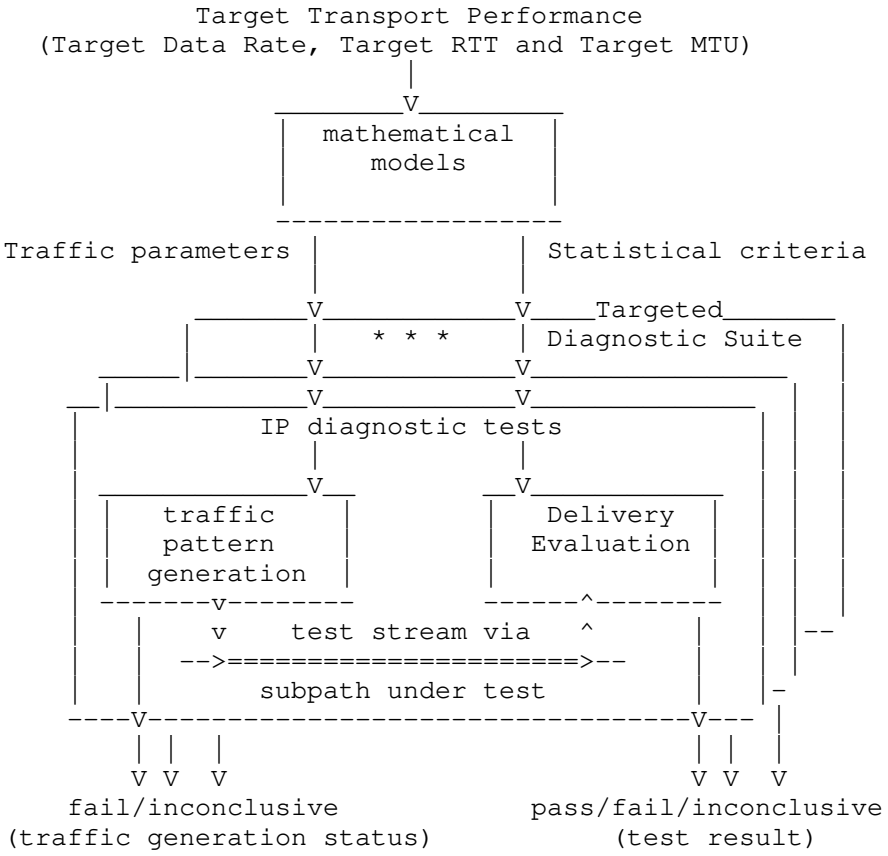
2. Overview

This document describes a modeling framework for deriving a Targeted IP Diagnostic Suite from a predetermined Target Transport Performance. It is not a complete specification, and relies on other standards documents to define important details such as packet Type-P selection, sampling techniques, vantage selection, etc. We imagine Fully Specified - Targeted IP Diagnostic Suites (FS-TIDS), that define all of these details. We use Targeted IP Diagnostic Suite (TIDS) to refer to the subset of such a specification that is in scope for this document. This terminology is defined in Section 3.

Section 4 describes some key aspects of TCP behavior and what they imply about the requirements for IP packet transfer. Most of the IP diagnostic tests needed to confirm that the path meets these properties can be built on existing IPPM metrics, with the addition of statistical criteria for evaluating packet transfer and in a few cases, new mechanisms to implement the required temporal structure. (One group of tests, the standing queue tests described in Section 8.2, don't correspond to existing IPPM metrics, but suitable new IPPM metrics can be patterned after the existing definitions.)

Figure 1 shows the MBM modeling and measurement framework. The Target Transport Performance, at the top of the figure, is determined by the needs of the user or application, outside the scope of this document. For Bulk Transport Capacity, the main performance parameter of interest is the Target Data Rate. However, since TCP's ability to compensate for less than ideal network conditions is fundamentally affected by the Round Trip Time (RTT) and the Maximum Transmission Unit (MTU) of the complete path, these parameters must also be specified in advance based on knowledge about the intended application setting. They may reflect a specific application over a real path through the Internet or an idealized application and

hypothetical path representing a typical user community. Section 5 describes the common parameters and models derived from the Target Transport Performance.



Overall Modeling Framework

Figure 1

Mathematical TCP models are used to determine Traffic parameters and subsequently to design traffic patterns that mimic TCP or other transport protocol delivering bulk data and operating at the Target Data Rate, MTU and RTT over a full range of conditions, including flows that are bursty at multiple time scales. The traffic patterns are generated based on the three Target parameters of complete path and independent of the properties of individual subpaths using the techniques described in Section 6. As much as possible the test streams are generated deterministically (precomputed) to minimize the extent to which test methodology, measurement points, measurement

vantage or path partitioning affect the details of the measurement traffic.

Section 7 describes packet transfer statistics and methods to test them against the statistical criteria provided by the mathematical models. Since the statistical criteria typically apply to the complete path (a composition of subpaths) [RFC6049], in situ testing requires that the end-to-end statistical criteria be apportioned as separate criteria for each subpath. Subpaths that are expected to be bottlenecks would then be permitted to contribute a larger fraction of the end-to-end packet loss budget. In compensation, subpaths that are not expected to exhibit bottlenecks must be constrained to contribute less packet loss. Thus the statistical criteria for each subpath in each test of a TIDS is an apportioned share of the end-to-end statistical criteria for the complete path which was determined by the mathematical model.

Section 8 describes the suite of individual tests needed to verify all of required IP delivery properties. A subpath passes if and only if all of the individual IP diagnostic tests pass. Any subpath that fails any test indicates that some users are likely to fail to attain their Target Transport Performance under some conditions. In addition to passing or failing, a test can be deemed to be inconclusive for a number of reasons including: the precomputed traffic pattern was not accurately generated; the measurement results were not statistically significant; and others such as failing to meet some required test preconditions. If all tests pass but some are inconclusive, then the entire suite is deemed to be inconclusive.

In Section 9 we present an example TIDS that might be representative of High Definition (HD) video, and illustrate how Model Based Metrics can be used to address difficult measurement situations, such as confirming that inter-carrier exchanges have sufficient performance and capacity to deliver HD video between ISPs.

Since there is some uncertainty in the modeling process, Section 10 describes a validation procedure to diagnose and minimize false positive and false negative results.

3. Terminology

Terms containing underscores (rather than spaces) appear in equations and typically have algorithmic definitions.

General Terminology:

Target: A general term for any parameter specified by or derived from the user's application or transport performance requirements.

Target Transport Performance: Application or transport performance target values for the complete path. For Bulk Transport Capacity defined in this note the Target Transport Performance includes the Target Data Rate, Target RTT and Target MTU as described below.

Target Data Rate: The specified application data rate required for an application's proper operation. Conventional Bulk Transport Capacity (BTC) metrics are focused on the Target Data Rate, however these metrics had little or no predictive value because they do not consider the effects of the other two parameters of the Target Transport Performance, the RTT and MTU of the complete paths.

Target RTT (Round Trip Time): The specified baseline (minimum) RTT of the longest complete path over which the user expects to be able to meet the target performance. TCP and other transport protocol's ability to compensate for path problems is generally proportional to the number of round trips per second. The Target RTT determines both key parameters of the traffic patterns (e.g. burst sizes) and the thresholds on acceptable IP packet transfer statistics. The Target RTT must be specified considering appropriate packets sizes: MTU sized packets on the forward path, ACK sized packets (typically header_overhead) on the return path. Note that Target RTT is specified and not measured, MBM measurements derived for a given target_RTT will be applicable to any path with a smaller RTTs.

Target MTU (Maximum Transmission Unit): The specified maximum MTU supported by the complete path the over which the application expects to meet the target performance. In this document assume a 1500 Byte MTU unless otherwise specified. If some subpath has a smaller MTU, then it becomes the Target MTU for the complete path, and all model calculations and subpath tests must use the same smaller MTU.

Targeted IP Diagnostic Suite (TIDS): A set of IP diagnostic tests designed to determine if an otherwise ideal complete path containing the subpath under test can sustain flows at a specific target_data_rate using target_MTU sized packets when the RTT of the complete path is target_RTT.

Fully Specified Targeted IP Diagnostic Suite (FS-TIDS): A TIDS together with additional specification such as measurement packet type ("type-p" [RFC2330]), etc. which are out of scope for this document, but need to be drawn from other standards documents.

Bulk Transport Capacity: Bulk Transport Capacity Metrics evaluate an Internet path's ability to carry bulk data, such as large files, streaming (non-real time) video, and under some conditions, web images and other content. Prior efforts to define BTC metrics have been based on [RFC3148], which predates our understanding of TCP and the requirements described in Section 4. In general "Bulk Transport" indicates that performance is determined by the interplay between the network, cross traffic and congestion

control in the transport protocol. It excludes situations where performance is dominated by the RTT alone (e.g. transactions) or bottlenecks elsewhere, such as in the application itself.

IP diagnostic tests: Measurements or diagnostics to determine if packet transfer statistics meet some precomputed target.

traffic patterns: The temporal patterns or burstiness of traffic generated by applications over transport protocols such as TCP. There are several mechanisms that cause bursts at various time scales as described in Section 4.1. Our goal here is to mimic the range of common patterns (burst sizes and rates, etc), without tying our applicability to specific applications, implementations or technologies, which are sure to become stale.

Explicit Congestion Notification (ECN): See [RFC3168].

packet transfer statistics: Raw, detailed or summary statistics about packet transfer properties of the IP layer including packet losses, ECN Congestion Experienced (CE) marks, reordering, or any other properties that may be germane to transport performance.

packet loss ratio: As defined in [RFC7680].

apportioned: To divide and allocate, for example budgeting packet loss across multiple subpaths such that the losses will accumulate to less than a specified end-to-end loss ratio. Apportioning metrics is essentially the inverse of the process described in [RFC5835].

open loop: A control theory term used to describe a class of techniques where systems that naturally exhibit circular dependencies can be analyzed by suppressing some of the dependencies, such that the resulting dependency graph is acyclic.

Terminology about paths, etc. See [RFC2330] and [RFC7398] for existing terms and definitions.

data sender: Host sending data and receiving ACKs.

data receiver: Host receiving data and sending ACKs.

complete path: The end-to-end path from the data sender to the data receiver.

subpath: A portion of the complete path. Note that there is no requirement that subpaths be non-overlapping. A subpath can be a small as a single device, link or interface.

measurement point: Measurement points as described in [RFC7398].

test path: A path between two measurement points that includes a subpath of the complete path under test. If the measurement points are off path, the test path may include "test leads" between the measurement points and the subpath.

dominant bottleneck: The bottleneck that generally determines most of packet transfer statistics for the entire path. It typically determines a flow's self clock timing, packet loss and ECN Congestion Experienced (CE) marking rate, with other potential

bottlenecks having less effect on the packet transfer statistics. See Section 4.1 on TCP properties.

front path: The subpath from the data sender to the dominant bottleneck.

back path: The subpath from the dominant bottleneck to the receiver.

return path: The path taken by the ACKs from the data receiver to the data sender.

cross traffic: Other, potentially interfering, traffic competing for network resources (bandwidth and/or queue capacity).

Properties determined by the complete path and application. These are described in more detail in Section 5.1.

Application Data Rate: General term for the data rate as seen by the application above the transport layer in bytes per second. This is the payload data rate, and explicitly excludes transport and lower level headers (TCP/IP or other protocols), retransmissions and other overhead that is not part to the total quantity of data delivered to the application.

IP rate: The actual number of IP-layer bytes delivered through a subpath, per unit time, including TCP and IP headers, retransmits and other TCP/IP overhead. Follows from IP-type-P Link Usage [RFC5136].

IP capacity: The maximum number of IP-layer bytes that can be transmitted through a subpath, per unit time, including TCP and IP headers, retransmits and other TCP/IP overhead. Follows from IP-type-P Link Capacity [RFC5136].

bottleneck IP capacity: The IP capacity of the dominant bottleneck in the forward path. All throughput maximizing protocols estimate this capacity by observing the IP rate delivered through the bottleneck. Most protocols derive their self clocks from the timing of this data. See Section 4.1 and Appendix B for more details.

implied bottleneck IP capacity: This is the bottleneck IP capacity implied by the ACKs returning from the receiver. It is determined by looking at how much application data the ACK stream at the sender reports delivered to the data receiver per unit time at various time scales. If the return path is thinning, batching or otherwise altering the ACK timing the implied bottleneck IP capacity over short time scales might be substantially larger than the bottleneck IP capacity averaged over a full RTT. Since TCP derives its clock from the data delivered through the bottleneck, the front path must have sufficient buffering to absorb any data bursts at the dimensions (size and IP rate) implied by the ACK stream, which are potentially doubled during slowstart. If the return path is not altering the ACK stream, then the implied bottleneck IP capacity will be the same as the bottleneck IP capacity. See Section 4.1 and Appendix B for more details.

sender interface rate: The IP rate which corresponds to the IP capacity of the data sender's interface. Due to sender efficiency algorithms including technologies such as TCP segmentation offload (TSO), nearly all modern servers deliver data in bursts at full interface link rate. Today 1 or 10 Gb/s are typical.

Header_overhead: The IP and TCP header sizes, which are the portion of each MTU not available for carrying application payload. Without loss of generality this is assumed to be the size for returning acknowledgments (ACKs). For TCP, the Maximum Segment Size (MSS) is the Target MTU minus the header_overhead.

Basic parameters common to models and subpath tests are defined here are described in more detail in Section 5.2. Note that these are mixed between application transport performance (excludes headers) and IP performance (which include TCP headers and retransmissions as part of the IP payload).

Network power: The observed data rate divided by the observed RTT. Network power indicates how effectively a transport protocol is filling a network.

Window [size]: The total quantity of data carried by packets in-flight plus the data represented by ACKs circulating in the network is referred to as the window. See Section 4.1. Sometimes used with other qualifiers (congestion window, cwnd or receiver window) to indicate which mechanism is controlling the window.

pipe size: A general term for number of packets needed in flight (the window size) to exactly fill some network path or subpath. It corresponds to the window size which maximizes network power. Often used with additional qualifiers to specify which path, or under what conditions, etc.

target_window_size: The average number of packets in flight (the window size) needed to meet the Target Data Rate, for the specified Target RTT, and MTU. It implies the scale of the bursts that the network might experience.

run length: A general term for the observed, measured, or specified number of packets that are (expected to be) delivered between losses or ECN Congestion Experienced (CE) marks. Nominally one over the sum of the loss and ECN CE marking probabilities, if there are independently and identically distributed.

target_run_length: The target_run_length is an estimate of the minimum number of non-congestion marked packets needed between losses or ECN Congestion Experienced (CE) marks necessary to attain the target_data_rate over a path with the specified target_RTT and target_MTU, as computed by a mathematical model of TCP congestion control. A reference calculation is shown in Section 5.2 and alternatives in Appendix A

reference target_run_length: target_run_length computed precisely by the method in Section 5.2. This is likely to be slightly more conservative than required by modern TCP implementations.

Ancillary parameters used for some tests:

derating: Under some conditions the standard models are too conservative. The modeling framework permits some latitude in relaxing or "derating" some test parameters as described in Section 5.3 in exchange for a more stringent TIDS validation procedures, described in Section 10. Models can be derated by including a multiplicative derating factor to make tests less stringent.

subpath_IP_capacity: The IP capacity of a specific subpath.

test path: A subpath of a complete path under test.

test_path_RTT: The RTT observed between two measurement points using packet sizes that are consistent with the transport protocol. This is generally MTU sized packets of the forward path, header_overhead sized packets on the return path.

test_path_pipe: The pipe size of a test path. Nominally the test_path_RTT times the test path IP_capacity.

test_window: The smallest window sufficient to meet or exceed the target_rate when operating with a pure self clock over a test path. The test_window is typically given by ceiling(target_data_rate*test_path_RTT/(target_MTU-header_overhead)) but see the discussion in Appendix B about the effects of channel scheduling on RTT. On some test paths the test_window may need to be adjusted slightly to compensate for the RTT being inflated by the devices that schedule packets.

The terminology below is used to define temporal patterns for test stream. These patterns are designed to mimic TCP behavior, as described in Section 4.1.

packet headway: Time interval between packets, specified from the start of one to the start of the next. e.g. If packets are sent with a 1 mS headway, there will be exactly 1000 packets per second.

burst headway: Time interval between bursts, specified from the start of the first packet one burst to the start of the first packet of the next burst. e.g. If 4 packet bursts are sent with a 1 mS burst headway, there will be exactly 4000 packets per second.

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

paced bursts: Send bursts on a timer. Specify any 3 of: average data rate, packet size, burst size (number of packets) and burst headway (burst start to start). By default the bursts are assumed to occur at full sender interface rate, such that the packet

headway within each burst is the minimum supported by the sender's interface. Under some conditions it is useful to explicitly specify the packet headway within each burst.

slowstart rate: Mimic TCP slowstart by sending 4 packet paced bursts at an average data rate equal to twice the implied bottleneck IP capacity (but not more than the sender interface rate). This is a two level burst pattern described in more detail in Section 6.1. If the implied bottleneck IP capacity is more than half of the sender interface rate, slowstart rate becomes sender interface rate.

slowstart burst: Mimic one round of TCP slowstart by sending a specified number of packets packets in a two level burst pattern that resembles slowstart.

repeated slowstart bursts: Repeat Slowstart bursts once per target_RTT. For TCP each burst would be twice as large as the prior burst, and the sequence would end at the first ECN CE mark or lost packet. For measurement, all slowstart bursts would be the same size (nominally target_window_size but other sizes might be specified), and the ECN CE marks and lost packets are counted.

The tests described in this note can be grouped according to their applicability.

Capacity tests: Capacity tests determine if a network subpath has sufficient capacity to deliver the Target Transport Performance. As long as the test stream is within the proper envelope for the Target Transport Performance, the average packet losses or ECN Congestion Experienced (CE) marks must be below the statistical criteria computed by the model. As such, capacity tests reflect parameters that can transition from passing to failing as a consequence of cross traffic, additional presented load or the actions of other network users. By definition, capacity tests also consume significant network resources (data capacity and/or queue buffer space), and the test schedules must be balanced by their cost.

Monitoring tests: Monitoring tests are designed to capture the most important aspects of a capacity test, but without presenting excessive ongoing load themselves. As such they may miss some details of the network's performance, but can serve as a useful reduced-cost proxy for a capacity test, for example to support continuous production network monitoring.

Engineering tests: Engineering tests evaluate how network algorithms (such as AQM and channel allocation) interact with TCP-style self clocked protocols and adaptive congestion control based on packet loss and ECN Congestion Experienced (CE) marks. These tests are likely to have complicated interactions with cross traffic and under some conditions can be inversely sensitive to load. For example a test to verify that an AQM algorithm causes ECN CE marks

or packet drops early enough to limit queue occupancy may experience a false pass result in the presence of cross traffic. It is important that engineering tests be performed under a wide range of conditions, including both in situ and bench testing, and over a wide variety of load conditions. Ongoing monitoring is less likely to be useful for engineering tests, although sparse in situ testing might be appropriate.

4. Background

At the time the "Framework for IP Performance Metrics" [RFC2330] was published (1998), sound Bulk Transport Capacity (BTC) measurement was known to be well beyond our capabilities. Even when Framework for Empirical BTC Metrics [RFC3148] was published, we knew that we didn't really understand the problem. Now, by hindsight we understand why assessing BTC is such a hard problem:

- TCP is a control system with circular dependencies - everything affects performance, including components that are explicitly not part of the test (for example, the host processing power is not in-scope of path performance tests).
- Congestion control is a dynamic equilibrium process, similar to processes observed in chemistry and other fields. The network and transport protocols find an operating point which balances between opposing forces: the transport protocol pushing harder (raising the data rate and/or window) while the network pushes back (raising packet loss ratio, RTT and/or ECN CE marks). By design TCP congestion control keeps raising the data rate until the network gives some indication that its capacity has been exceeded by dropping packets or adding ECN CE marks. If a TCP sender accurately fills a path to its IP capacity, (e.g. the bottleneck is 100% utilized), then packet losses and ECN CE marks are mostly determined by the TCP sender and how aggressively it seeks additional capacity, and not the network itself, since the network must send exactly the signals that TCP needs to set its rate.
- TCP's ability to compensate for network impairments (such as loss, delay and delay variation, outside of those caused by TCP itself) is directly proportional to the number of send-ACK round trip exchanges per second (i.e. inversely proportional to the RTT). As a consequence an impaired subpath may pass a short RTT local test even though it fails when the subpath is extended by an effectively perfect network to some larger RTT.
- TCP has an extreme form of the Observer Effect (colloquially know as the Heisenberg effect). Measurement and cross traffic interact in unknown and ill defined ways. The situation is actually worse than the traditional physics problem where you can at least estimate bounds on the relative momentum of the measurement and measured particles. For network measurement you can not in

general determine even the order of magnitude of the effect. It is possible to construct measurement scenarios where the measurement traffic starves real user traffic, yielding an overly inflated measurement. The inverse is also possible: the user traffic can fill the network, such that the measurement traffic detects only minimal available capacity. You can not in general determine which scenario might be in effect, so you can not gauge the relative magnitude of the uncertainty introduced by interactions with other network traffic.

- o As a consequence of the properties listed above it is difficult, if not impossible, for two independent implementations (HW or SW) of TCP congestion control to produce equivalent performance results [RFC6576] under the same network conditions,

These properties are a consequence of the dynamic equilibrium behavior intrinsic to how all throughput maximizing protocols interact with the Internet. These protocols rely on control systems based on estimated network metrics to regulate the quantity of data to send into the network. The packet sending characteristics in turn alter the network properties estimated by the control system metrics, such that there are circular dependencies between every transmission characteristic and every estimated metric. Since some of these dependencies are nonlinear, the entire system is nonlinear, and any change anywhere causes a difficult to predict response in network metrics. As a consequence Bulk Transport Capacity metrics have not fulfilled the analytic framework envisioned in [RFC2330]

Model Based Metrics overcome these problems by making the measurement system open loop: the packet transfer statistics (akin to the network estimators) do not affect the traffic or traffic patterns (bursts), which are computed on the basis of the Target Transport Performance. A path or subpath meeting the Target Transfer Performance requirements would exhibit packet transfer statistics and estimated metrics that would not cause the control system to slow the traffic below the Target Data Rate.

4.1. TCP properties

TCP and other self clocked protocols (e.g. SCTP) carry the vast majority of all Internet data. Their dominant bulk data transport behavior is to have an approximately fixed quantity of data and acknowledgments (ACKs) circulating in the network. The data receiver reports arriving data by returning ACKs to the data sender, the data sender typically responds by sending approximately the same quantity of data back into the network. The total quantity of data plus the data represented by ACKs circulating in the network is referred to as the window. The mandatory congestion control algorithms incrementally adjust the window by sending slightly more or less data

in response to each ACK. The fundamentally important property of this system is that it is self clocked: The data transmissions are a reflection of the ACKs that were delivered by the network, the ACKs are a reflection of the data arriving from the network.

A number of protocol features cause bursts of data, even in idealized networks that can be modeled as simple queuing systems.

During slowstart the IP rate is doubled on each RTT by sending twice as much data as was delivered to the receiver during the prior RTT. Each returning ACK causes the sender to transmit twice the data the ACK reported arriving at the receiver. For slowstart to be able to fill the pipe, the network must be able to tolerate slowstart bursts up to the full pipe size inflated by the anticipated window reduction on the first loss or ECN CE mark. For example, with classic Reno congestion control, an optimal slowstart has to end with a burst that is twice the bottleneck rate for one RTT in duration. This burst causes a queue which is equal to the pipe size (i.e. the window is twice the pipe size) so when the window is halved in response to the first packet loss, the new window will be the pipe size.

Note that if the bottleneck IP rate is less that half of the capacity of the front path (which is almost always the case), the slowstart bursts will not by themselves cause significant queues anywhere else along the front path; they primarily exercise the queue at the dominant bottleneck.

Several common efficiency algorithms also cause bursts. The self clock is typically applied to groups of packets: the receiver's delayed ACK algorithm generally sends only one ACK per two data segments. Furthermore the modern senders use TCP segmentation offload (TSO) to reduce CPU overhead. The sender's software stack builds super sized TCP segments that the TSO hardware splits into MTU sized segments on the wire. The net effect of TSO, delayed ACK and other efficiency algorithms is to send bursts of segments at full sender interface rate.

Note that these efficiency algorithms are almost always in effect, including during slowstart, such that slowstart typically has a two level burst structure. Section 6.1 describes slowstart in more detail.

Additional sources of bursts include TCP's initial window [RFC6928], application pauses, channel allocation mechanisms and network devices that schedule ACKs. Appendix B describes these last two items. If the application pauses (stops reading or writing data) for some fraction of an RTT, many TCP implementations catch up to their earlier window size by sending a burst of data at the full sender

interface rate. To fill a network with a realistic application, the network has to be able to tolerate sender interface rate bursts large enough to restore the prior window following application pauses.

Although the sender interface rate bursts are typically smaller than the last burst of a slowstart, they are at a higher IP rate so they potentially exercise queues at arbitrary points along the front path from the data sender up to and including the queue at the dominant bottleneck. It is known that these bursts can hurt network performance, especially in conjunction with other queue pressure, however we are not aware of any models for how frequent sender rate bursts the network should be able to tolerate at various burst sizes.

In conclusion, to verify that a path can meet a Target Transport Performance, it is necessary to independently confirm that the path can tolerate bursts at the scales that can be caused by the above mechanisms. Three cases are believed to be sufficient:

- Two level slowstart bursts sufficient to get connections started properly.
- Ubiquitous sender interface rate bursts caused by efficiency algorithms. We assume 4 packet bursts to be the most common case, since it matches the effects of delayed ACK during slowstart. These bursts should be assumed not to significantly affect packet transfer statistics.
- Infrequent sender interface rate bursts that are the maximum of the full target_window_size and the initial window size (10 segments in [RFC6928]). The Target_run_length may be derated for these large fast bursts.

If a subpath can meet the required packet loss ratio for bursts at all of these scales then it has sufficient buffering at all potential bottlenecks to tolerate any of the bursts that are likely introduced by TCP or other transport protocols.

4.2. Diagnostic Approach

A complete path of a given RTT and MTU, which are equal to or smaller than the Target RTT and equal to or larger than the Target MTU respectively, is expected to be able to attain a specified Bulk Transport Capacity when all of the following conditions are met:

1. The IP capacity is above the Target Data Rate by sufficient margin to cover all TCP/IP overheads. This can be confirmed by the tests described in Section 8.1 or any number of IP capacity tests adapted to implement MBM.
2. The observed packet transfer statistics are better than required by a suitable TCP performance model (e.g. fewer packet losses or

ECN CE marks). See Section 8.1 or any number of low or fixed rate packet loss tests outside of MBM.

3. There is sufficient buffering at the dominant bottleneck to absorb a slowstart bursts large enough to get the flow out of slowstart at a suitable window size. See Section 8.3.
4. There is sufficient buffering in the front path to absorb and smooth sender interface rate bursts at all scales that are likely to be generated by the application, any channel arbitration in the ACK path or any other mechanisms. See Section 8.4.
5. When there is a slowly rising standing queue at the bottleneck the onset of packet loss has to be at an appropriate point (time or queue depth) and progressive [RFC7567]. See Section 8.2.
6. When there is a standing queue at a bottleneck for a shared media subpath (e.g. half duplex), there must be a suitable bounds on the interaction between ACKs and data, for example due to the channel arbitration mechanism. See Section 8.2.4.

Note that conditions 1 through 4 require capacity tests for validation, and thus may need to be monitored on an ongoing basis. Conditions 5 and 6 require engineering tests, which are best performed in controlled environments such as a bench test. They won't generally fail due to load, but may fail in the field due to configuration errors, etc. and should be spot checked.

A tool that can perform many of the tests is available from [MBMSource].

4.3. New requirements relative to RFC 2330

Model Based Metrics are designed to fulfill some additional requirements that were not recognized at the time RFC 2330 was written [RFC2330]. These missing requirements may have significantly contributed to policy difficulties in the IP measurement space. Some additional requirements are:

- 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.
- Metrics must be vantage point invariant over a significant range of measurement point choices, including off path measurement points. The only requirements on MP selection should be that the RTT between the MPs is below some reasonable bound, and that the effects of the "test leads" connecting MPs to the subpath under test can be can be calibrated out of the measurements. The latter might be be accomplished if the test leads are effectively ideal or their properties can be deducted from the measurements between

the MPs. While many of tests require that the test leads have at least as much IP capacity as the subpath under test, some do not, for example Background Packet Transfer Tests described in Section 8.1.3.

- o Metric measurements should be repeatable by multiple parties with no specialized access to MPs or diagnostic infrastructure. It should be possible for different parties to make the same measurement and observe the same results. In particular it is specifically important that both a consumer (or their delegate) and ISP be able to perform the same measurement and get the same result. Note that vantage independence is key to meeting this requirement.

5. Common Models and Parameters

5.1. Target End-to-end parameters

The target end-to-end parameters are the Target Data Rate, Target RTT and Target MTU as defined in Section 3. These parameters are determined by the needs of the application or the ultimate end user and the complete Internet path over which the application is expected to operate. The target parameters are in units that make sense to upper layers: payload bytes delivered to the application, above TCP. They exclude overheads associated with TCP and IP headers, retransmits and other protocols (e.g. DNS). Note that IP-based network services include TCP headers and retransmissions as part of delivered payload, and this difference is recognized in calculations below (header_overhead).

Other end-to-end parameters defined in Section 3 include the effective bottleneck data rate, the sender interface data rate and the TCP and IP header sizes.

The target_data_rate must be smaller than all subpath IP capacities by enough headroom to carry the transport protocol overhead, explicitly including retransmissions and an allowance for fluctuations in TCP's actual data rate. Specifying a target_data_rate with insufficient headroom is likely to result in brittle measurements having little predictive value.

Note that the target parameters can be specified for a hypothetical path, for example to construct TIDS designed for bench testing in the absence of a real application; or for a live in situ test of production infrastructure.

The number of concurrent connections is explicitly not a parameter to this model. If a subpath requires multiple connections in order to

meet the specified performance, that must be stated explicitly and the procedure described in Section 6.4 applies.

5.2. Common Model Calculations

The Target Transport Performance is used to derive the target_window_size and the reference target_run_length.

The target_window_size, is the average window size in packets needed to meet the target_rate, for the specified target_RTT and target_MTU. It is given by:

target_window_size = ceiling(target_rate * target_RTT / (target_MTU - header_overhead))

Target_run_length is an estimate of the minimum required number of unmarked packets that must be delivered between losses or ECN Congestion Experienced (CE) marks, as computed by a mathematical model of TCP congestion control. The derivation here follows [MSMO97], and by design is quite conservative.

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

Assume the transport protocol is using standard Reno style Additive Increase, Multiplicative Decrease (AIMD) congestion control [RFC5681] (but not Appropriate Byte Counting [RFC3465]) and the receiver is using standard delayed ACKs. Reno increases the window by one packet every pipe_size worth of ACKs. With delayed ACKs this takes 2 Round Trip Times per increase. To exactly fill the pipe, the spacing of losses must be no closer than when the peak of the AIMD sawtooth reached exactly twice the target_window_size. Otherwise, the multiplicative window reduction triggered by the loss would cause the network to be under-filled. Following [MSMO97] the number of packets between losses must be the area under the AIMD sawtooth. They must be no more frequent than every 1 in ((3/2)*target_window_size)*(2*target_window_size) packets, which simplifies to:

target_run_length = 3*(target_window_size^2)

Note that this calculation is very conservative and is based on a number of assumptions that may not apply. Appendix A discusses these assumptions and provides some alternative models. If a different model is used, a FS-TIDS must document the actual method for

computing target_run_length and ratio between alternate target_run_length and the reference target_run_length calculated above, along with a discussion of the rationale for the underlying assumptions.

These two parameters, target_window_size and target_run_length, directly imply most of the individual parameters for the tests in Section 8.

5.3. Parameter Derating

Since some aspects of the models are very conservative, the MBM framework permits some latitude in derating test parameters. Rather than trying to formalize more complicated models we permit some test parameters to be relaxed as long as they meet some additional procedural constraints:

- The FS-TIDS must document and justify the actual method used to compute the derated metric parameters.
- The validation procedures described in Section 10 must be used to demonstrate the feasibility of meeting the Target Transport Performance with infrastructure that just barely passes the derated tests.
- The validation process for a FS-TIDS itself must be documented is such a way that other researchers can duplicate the validation experiments.

Except as noted, all tests below assume no derating. Tests where there is not currently a well established model for the required parameters explicitly include derating as a way to indicate flexibility in the parameters.

5.4. Test Preconditions

Many tests have preconditions which are required to assure their validity. Examples include: the presence or non-presence of cross traffic on specific subpaths; negotiating ECN; and appropriate preamble packet stream to testing to put reactive network elements into the proper states [RFC7312]. If preconditions are not properly satisfied for some reason, the tests should be considered to be inconclusive. In general it is useful to preserve diagnostic information as to why the preconditions were not met, and any test data that was collected even if it is not useful for the intended test. Such diagnostic information and partial test data may be useful for improving the test or test procedures themselves.

It is important to preserve the record that a test was scheduled, because otherwise precondition enforcement mechanisms can introduce

sampling bias. For example, canceling tests due to cross traffic on subscriber access links might introduce sampling bias in tests of the rest of the network by reducing the number of tests during peak network load.

Test preconditions and failure actions must be specified in a FS-TIDS.

6. Generating test streams

Many important properties of Model Based Metrics, such as vantage independence, are a consequence of using test streams that have temporal structures that mimic TCP or other transport protocols running over a complete path. As described in Section 4.1, self clocked protocols naturally have burst structures related to the RTT and pipe size of the complete path. These bursts naturally get larger (contain more packets) as either the Target RTT or Target Data Rate get larger, or the Target MTU gets smaller. An implication of these relationships is that test streams generated by running self clocked protocols over short subpaths may not adequately exercise the queuing at any bottleneck to determine if the subpath can support the full Target Transport Performance over the complete path.

Failing to authentically mimic TCP's temporal structure is part of the reason why simple performance tools such as iPerf, netperf, nc, etc have the reputation of yielding false pass results over short test paths, even when some subpath has a flaw.

The definitions in Section 3 are sufficient for most test streams. We describe the slowstart and standing queue test streams in more detail.

In conventional measurement practice stochastic processes are used to eliminate many unintended correlations and sample biases. However MBM tests are designed to explicitly mimic temporal correlations caused by network or protocol elements themselves. Some portions of these systems, such as traffic arrival (test scheduling) are naturally stochastic. Other behaviors, such as back-to-back packet transmissions, are dominated by implementation specific deterministic effects. Although these behaviors always contain non-deterministic elements and might be modeled stochastically, these details typically do not contribute significantly to the overall system behavior. Furthermore, it is known that real protocols are subject to failures caused by network property estimators suffering from bias due to correlation in their own traffic. For example TCP's RTT estimator used to determine the Retransmit Time Out (RTO), can be fooled by periodic cross traffic or start-stop applications. For these reasons many details of the test streams are specified deterministically.

It may prove useful to introduce fine grained noise sources into the models used for generating test streams in an update of Model Based Metrics, but the complexity is not warranted at the time this document was written.

6.1. Mimicking slowstart

TCP slowstart has a two level burst structure as shown in Figure 2. The fine time structure is caused by efficiency algorithms that deliberately batch work (CPU, channel allocation, etc) to better amortize certain network and host overheads. ACKs passing through the return path typically cause the sender to transmit small bursts of data at full sender interface rate. For example TCP Segmentation Offload (TSO) and Delayed Acknowledgment both contribute to this effect. During slowstart these bursts are at the same headway as the returning ACKs, but are typically twice as large (e.g. having twice as much data) as the ACK reported was delivered to the receiver. Due to variations in delayed ACK and algorithms such as Appropriate Byte Counting [RFC3465], different pairs of senders and receivers produce slightly different burst patterns. Without loss of generality, we assume each ACK causes 4 packet sender interface rate bursts at an average headway equal to the ACK headway, and corresponding to sending at an average rate equal to twice the effective bottleneck IP rate. Each slowstart burst consists of a series of 4 packet sender interface rate bursts such that the total number of packets is the current window size (as of the last packet in the burst).

The coarse time structure is due to each RTT being a reflection of the prior RTT. For real transport protocols, each slowstart burst is twice as large (twice the window) as the previous burst but is spread out in time by the network bottleneck, such that each successive RTT exhibits the same effective bottleneck IP rate. The slowstart phase ends on the first lost packet or ECN mark, which is intended to happen after successive slowstart bursts merge in time: the next burst starts before the bottleneck queue is fully drained and the prior burst is complete.

For diagnostic tests described below we preserve the fine time structure but manipulate the coarse structure of the slowstart bursts (burst size and headway) to measure the ability of the dominant bottleneck to absorb and smooth slowstart bursts.

Note that a stream of repeated slowstart bursts has three different average rates, depending on the averaging time interval. At the finest time scale (a few packet times at the sender interface) the peak of the average IP rate is the same as the sender interface rate; at a medium timescale (a few ACK times at the dominant bottleneck) the peak of the average IP rate is twice the implied bottleneck IP

capacity; and at time scales longer than the target_RTT and when the burst size is equal to the target_window_size, the average rate is equal to the target_data_rate. This pattern corresponds to repeating the last RTT of TCP slowstart when delayed ACK and sender side byte counting are present but without the limits specified in Appropriate Byte Counting [RFC3465].

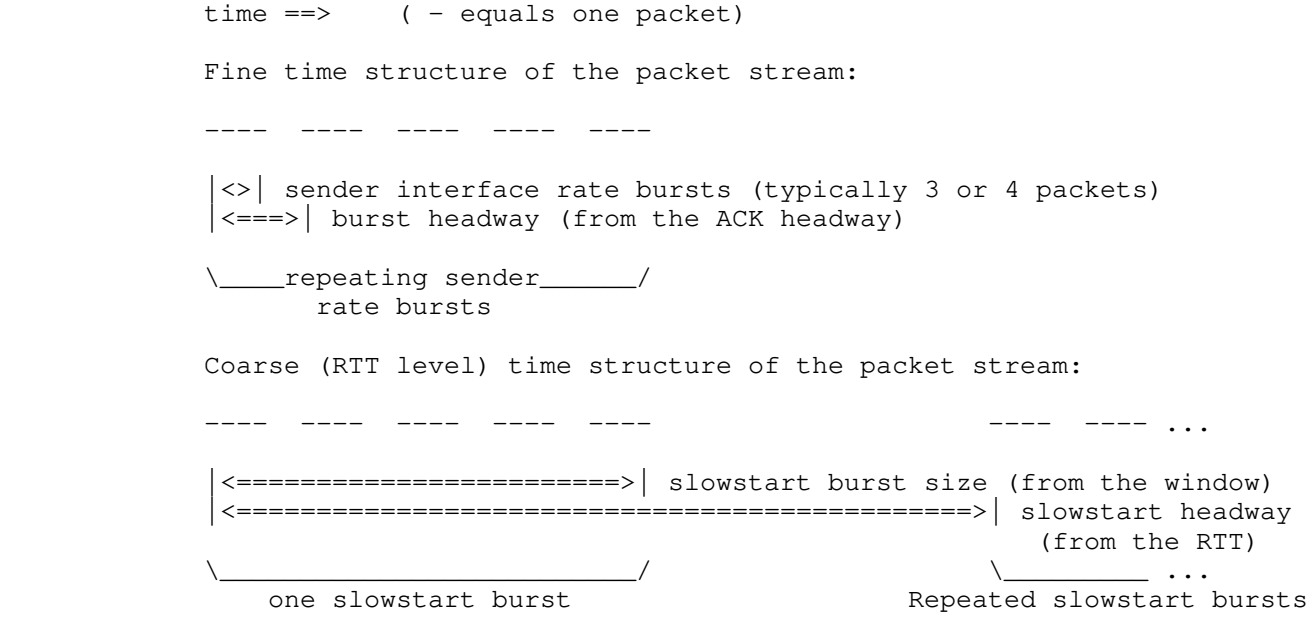


Figure 2

6.2. Constant window pseudo CBR

Implement pseudo constant bit rate by running a standard self clocked protocol such as TCP with a fixed window size. If that window size is test_window, the data rate will be slightly above the target_rate.

Since the test_window is constrained to be an integer number of packets, for small RTTs or low data rates there may not be sufficiently precise control over the data rate. Rounding the test_window up (as defined above) is likely to result in data rates that are higher than the target rate, but reducing the window by one packet may result in data rates that are too small. Also cross traffic potentially raises the RTT, implicitly reducing the rate.

Cross traffic that raises the RTT nearly always makes the test more strenuous (more demanding for the network path).

Note that Constant window pseudo CBR (and Scanned window pseudo CBR in the next section) both rely on a self clock which is at least partially derived from the properties of the subnet under test. This introduces the possibility that the subnet under test exhibits behaviors such as extreme RTT fluctuations that prevent these algorithms from accurately controlling data rates.

A FS-TIDS specifying a constant window CBR test must explicitly indicate under what conditions errors in the data rate cause tests to be inconclusive. Conventional paced measurement traffic may be more appropriate for these environments.

6.3. Scanned window pseudo CBR

Scanned window pseudo CBR is similar to the constant window CBR described above, except the window is scanned across a range of sizes designed to include two key events, the onset of queuing and the onset of packet loss or ECN CE marks. The window is scanned by incrementing it by one packet every 2*target_window_size delivered packets. This mimics the additive increase phase of standard Reno TCP congestion avoidance when delayed ACKs are in effect. Normally the window increases separated by intervals slightly longer than twice the target_RTT.

There are two ways to implement this test: one built by applying a window clamp to standard congestion control in a standard protocol such as TCP and the other built by stiffening a non-standard transport protocol. When standard congestion control is in effect, any losses or ECN CE marks cause the transport to revert to a window smaller than the clamp such that the scanning clamp loses control the window size. The NPAD pathdiag tool is an example of this class of algorithms [Pathdiag].

Alternatively a non-standard congestion control algorithm can respond to losses by transmitting extra data, such that it maintains the specified window size independent of losses or ECN CE marks. Such a stiffened transport explicitly violates mandatory Internet congestion control [RFC5681] and is not suitable for in situ testing. It is only appropriate for engineering testing under laboratory conditions. The Windowed Ping tool implements such a test [WPING]. The tool described in the paper has been updated.[mpingSource]

The test procedures in Section 8.2 describe how to the partition the scans into regions and how to interpret the results.

6.4. Concurrent or channelized testing

The procedures described in this document are only directly applicable to single stream measurement, e.g. one TCP connection or measurement stream. In an ideal world, we would disallow all performance claims based multiple concurrent streams, but this is not practical due to at least two issues. First, many very high rate link technologies are channelized and at last partially pin the flow to channel mapping to minimize packet reordering within flows. Second, TCP itself has scaling limits. Although the former problem might be overcome through different design decisions, the later problem is more deeply rooted.

All congestion control algorithms that are philosophically aligned with the standard [RFC5681] (e.g. claim some level of TCP compatibility, friendliness or fairness) have scaling limits, in the sense that as a long fast network (LFN) with a fixed RTT and MTU gets faster, these congestion control algorithms get less accurate and as a consequence have difficulty filling the network [CCscaling]. These properties are a consequence of the original Reno AIMD congestion control design and the requirement in [RFC5681] that all transport protocols have similar responses to congestion.

There are a number of reasons to want to specify performance in terms of multiple concurrent flows, however this approach is not recommended for data rates below several megabits per second, which can be attained with run lengths under 10000 packets on many paths. Since the required run length goes as the square of the data rate, at higher rates the run lengths can be unreasonably large, and multiple flows might be the only feasible approach.

If multiple flows are deemed necessary to meet aggregate performance targets then this must be stated in both the design of the TIDS and in any claims about network performance. The IP diagnostic tests must be performed concurrently with the specified number of connections. For the tests that use bursty test streams, the bursts should be synchronized across streams unless there is a priori knowledge that the applications have some explicit mechanism to stagger their own bursts. In the absences of an explicit mechanism to stagger bursts many network and application artifacts will sometimes implicitly synchronize bursts. A test that does not control burst synchronization may be prone to false pass results for some applications.

7. Interpreting the Results

7.1. Test outcomes

To perform an exhaustive test of a complete network path, each test of the TIDS is applied to each subpath of the complete path. If any subpath fails any test then a standard transport protocol running over the complete path can also be expected to fail to attain the Target Transport Performance under some conditions.

In addition to passing or failing, a test can be deemed to be inconclusive for a number of reasons. Proper instrumentation and treatment of inconclusive outcomes is critical to the accuracy and robustness of Model Based Metrics. Tests can be inconclusive if the precomputed traffic pattern or data rates were not accurately generated; the measurement results were not statistically significant; and others causes such as failing to meet some required preconditions for the test. See Section 5.4

For example consider a test that implements Constant Window Pseudo CBR (Section 6.2) by adding rate controls and detailed IP packet transfer instrumentation to TCP (e.g. [RFC4898]). TCP includes built in control systems which might interfere with the sending data rate. If such a test meets the required packet transfer statistics (e.g. run length) while failing to attain the specified data rate it must be treated as an inconclusive result, because we can not a priori determine if the reduced data rate was caused by a TCP problem or a network problem, or if the reduced data rate had a material effect on the observed packet transfer statistics.

Note that for capacity tests, if the observed packet transfer statistics meet the statistical criteria for failing (accepting hypnosis H1 in Section 7.2), the test can can be considered to have failed because it doesn't really matter that the test didn't attain the required data rate.

The really important new properties of MBM, such as vantage independence, are a direct consequence of opening the control loops in the protocols, such that the test stream does not depend on network conditions or IP packets received. Any mechanism that introduces feedback between the path's measurements and the test stream generation is at risk of introducing nonlinearities that spoil these properties. Any exceptional event that indicates that such feedback has happened should cause the test to be considered inconclusive.

One way to view inconclusive tests is that they reflect situations where a test outcome is ambiguous between limitations of the network

and some unknown limitation of the IP diagnostic test itself, which may have been caused by some uncontrolled feedback from the network.

Note that procedures that attempt to search the target parameter space to find the limits on some parameter such as target_data_rate are at risk of breaking the location independent properties of Model Based Metrics, if any part of the boundary between passing and inconclusive or failing results is sensitive to RTT (which is normally the case). For example the maximum data rate for a marginal link (e.g. exhibiting excess errors) is likely to be sensitive to the test_path_RTT. The maximum observed data rate over the test path has very little value for predicting the maximum rate over a different path.

One of the goals for evolving TIDS designs will be to keep sharpening distinction between inconclusive, passing and failing tests. The criteria for for passing, failing and inconclusive tests must be explicitly stated for every test in the TIDS or FS-TIDS.

One of the goals of evolving the testing process, procedures, tools and measurement point selection should be to minimize the number of inconclusive tests.

It may be useful to keep raw packet transfer statistics and ancillary metrics [RFC3148] for deeper study of the behavior of the network path and to measure the tools themselves. Raw packet transfer statistics can help to drive tool evolution. Under some conditions it might be possible to re-evaluate the raw data for satisfying alternate Target Transport Performance. However it is important to guard against sampling bias and other implicit feedback which can cause false results and exhibit measurement point vantage sensitivity. Simply applying different delivery criteria based on a different Target Transport Performance is insufficient if the test traffic patterns (bursts, etc.) does not match the alternate Target Transport Performance.

7.2. Statistical criteria for estimating run_length

When evaluating the observed run_length, we need to determine appropriate packet stream sizes and acceptable error levels for efficient measurement. In practice, can we compare the empirically estimated packet loss and ECN Congestion Experienced (CE) marking ratios with the targets as the sample size grows? How large a sample is needed to say that the measurements of packet transfer indicate a particular run length is present?

The generalized measurement can be described as recursive testing: send packets (individually or in patterns) and observe the packet

transfer performance (packet loss ratio or other metric, any marking we define).

As each packet is sent and measured, we have an ongoing estimate of the performance in terms of the ratio of packet loss or ECN CE mark to total packets (i.e. an empirical probability). We continue to send until conditions support a conclusion or a maximum sending limit has been reached.

We have a target_mark_probability, 1 mark per target_run_length, where a "mark" is defined as a lost packet, a packet with ECN CE mark, or other signal. This constitutes the null Hypothesis:

H0: no more than one mark in target_run_length = 3*(target_window_size)^2 packets

and we can stop sending packets if on-going measurements support accepting H0 with the specified Type I error = alpha (= 0.05 for example).

We also have an alternative Hypothesis to evaluate: if performance is significantly lower than the target_mark_probability. Based on analysis of typical values and practical limits on measurement duration, we choose four times the H0 probability:

H1: one or more marks in (target_run_length/4) packets

and we can stop sending packets if measurements support rejecting H0 with the specified Type II error = beta (= 0.05 for example), thus preferring the alternate hypothesis H1.

H0 and H1 constitute the Success and Failure outcomes described elsewhere in the memo, and while the ongoing measurements do not support either hypothesis the current status of measurements is inconclusive.

The problem above is formulated to match the Sequential Probability Ratio Test (SPRT) [Wald45] and [Montgomery90]. Note that as originally framed the events under consideration were all manufacturing defects. In networking, ECN CE marks and lost packets are not defects but signals, indicating that the transport protocol should slow down.

The Sequential Probability Ratio Test also starts with a pair of hypothesis specified as above:

H0: p0 = one defect in target_run_length
H1: p1 = one defect in target_run_length/4

As packets are sent and measurements collected, the tester evaluates the cumulative defect count against two boundaries representing H0 Acceptance or Rejection (and acceptance of H1):

```
Acceptance line:  Xa = -h1 + s*n
Rejection line:  Xr = h2 + s*n
```

where n increases linearly for each packet sent and

```
h1 =  { log((1-alpha)/beta) }/k
h2 =  { log((1-beta)/alpha) }/k
k  =  log{ (p1(1-p0)) / (p0(1-p1)) }
s  =  [ log{ (1-p0)/(1-p1) } ]/k
```

for p0 and p1 as defined in the null and alternative Hypotheses statements above, and alpha and beta as the Type I and Type II errors.

The SPRT specifies simple stopping rules:

- Xa < defect_count(n) < Xr: continue testing
- defect_count(n) <= Xa: Accept H0
- defect_count(n) >= Xr: Accept H1

The calculations above are implemented in the R-tool for Statistical Analysis [Rtool] , in the add-on package for Cross-Validation via Sequential Testing (CVST) [CVST].

Using the equations above, we can calculate the minimum number of packets (n) needed to accept H0 when x defects are observed. For example, when x = 0:

```
Xa = 0  = -h1 + s*n
and  n = h1 / s
```

Note that the derivations in [Wald45] and [Montgomery90] differ. Montgomery's simplified derivation of SPRT may assume a Bernoulli processes, where the packet loss probabilities are independent and identically distributed, making the SPRT more accessible. Wald's seminal paper showed that this assumption is not necessary. It helps to remember that the goal of SPRT is not to estimate the value of the packet loss rate, but only whether or not the packet loss ratio is likely low enough (when we accept the H0 null hypothesis) yielding success; or too high (when we accept the H1 alternate hypothesis) yielding failure.

7.3. Reordering Tolerance

All tests must be instrumented for packet level reordering [RFC4737]. However, there is no consensus for how much reordering should be acceptable. Over the last two decades the general trend has been to make protocols and applications more tolerant to reordering (see for example [RFC4015]), in response to the gradual increase in reordering in the network. This increase has been due to the deployment of technologies such as multithreaded routing lookups and Equal Cost MultiPath (ECMP) routing. These techniques increase parallelism in network and are critical to enabling overall Internet growth to exceed Moore's Law.

Note that transport retransmission strategies can trade off reordering tolerance vs how quickly they can repair losses vs overhead from spurious retransmissions. In advance of new retransmission strategies we propose the following strawman: Transport protocols should be able to adapt to reordering as long as the reordering extent is not more than the maximum of one quarter window or 1 mS, whichever is larger. (These values come from experience prototyping Early Retransmit [RFC5827] and related algorithms. They agree with the values being proposed for "RACK: a time-based fast loss detection algorithm" [I-D.ietf-tcpm-rack].) Within this limit on reorder extent, there should be no bound on reordering density.

By implication, recording which is less than these bounds should not be treated as a network impairment. However [RFC4737] still applies: reordering should be instrumented and the maximum reordering that can be properly characterized by the test (because of the bound on history buffers) should be recorded with the measurement results.

Reordering tolerance and diagnostic limitations, such as the size of the history buffer used to diagnose packets that are way out-of-order, must be specified in a FSTIDS.

8. IP Diagnostic Tests

The IP diagnostic tests below are organized according to the technique used to generate the test stream as described in Section 6. All of the results are evaluated in accordance with Section 7, possibly with additional test specific critera.

We also introduce some combined tests which are more efficient when networks are expected to pass, but conflate diagnostic signatures when they fail.

8.1. Basic Data Rate and Packet Transfer Tests

We propose several versions of the basic data rate and packet transfer statistics test that differ in how the data rate is controlled. The data can be paced on a timer, or window controlled (and self clocked). The first two tests implicitly confirm that sub_path has sufficient raw capacity to carry the target_data_rate. They are recommended for relatively infrequent testing, such as an installation or periodic auditing process. The third, background packet transfer statistics, is a low rate test designed for ongoing monitoring for changes in subpath quality.

8.1.1. Delivery Statistics at Paced Full Data Rate

Confirm that the observed run length is at least the target_run_length while relying on timer to send data at the target_rate using the procedure described in in Section 6.1 with a burst size of 1 (single packets) or 2 (packet pairs).

The test is considered to be inconclusive if the packet transmission can not be accurately controlled for any reason.

RFC 6673 [RFC6673] is appropriate for measuring packet transfer statistics at full data rate.

8.1.2. Delivery Statistics at Full Data Windowed Rate

Confirm that the observed run length is at least the target_run_length while sending at an average rate approximately equal to the target_data_rate, by controlling (or clamping) the window size of a conventional transport protocol to test_window.

Since losses and ECN CE marks cause transport protocols to reduce their data rates, this test is expected to be less precise about controlling its data rate. It should not be considered inconclusive as long as at least some of the round trips reached the full target_data_rate without incurring losses or ECN CE marks. To pass this test the network must deliver target_window_size packets in target_RTT time without any losses or ECN CE marks at least once per two target_window_size round trips, in addition to meeting the run length statistical test.

8.1.3. Background Packet Transfer Statistics Tests

The background run length is a low rate version of the target target rate test above, designed for ongoing lightweight monitoring for changes in the observed subpath run length without disrupting users. It should be used in conjunction with one of the above full rate

tests because it does not confirm that the subpath can support raw data rate.

RFC 6673 [RFC6673] is appropriate for measuring background packet transfer statistics.

8.2. Standing Queue Tests

These engineering tests confirm that the bottleneck is well behaved across the onset of packet loss, which typically follows after the onset of queuing. Well behaved generally means lossless for transient queues, but once the queue has been sustained for a sufficient period of time (or reaches a sufficient queue depth) there should be a small number of losses or ECN CE marks to signal to the transport protocol that it should reduce its window or data rate. Losses that are too early can prevent the transport from averaging at the target_data_rate. Losses that are too late indicate that the queue might not have an appropriate AQM [RFC7567] and as a consequence subject to bufferbloat [wikiBloat]. Queues without AQM have the potential to inflict excess delays on all flows sharing the bottleneck. Excess losses (more than half of the window) at the onset of loss make loss recovery problematic for the transport protocol. Non-linear, erratic or excessive RTT increases suggest poor interactions between the channel acquisition algorithms and the transport self clock. All of the tests in this section use the same basic scanning algorithm, described here, but score the link or subpath on the basis of how well it avoids each of these problems.

Some network technologies rely on virtual queues or other techniques to meter traffic without adding any queuing delay, in which case the data rate will vary with the window size all the way up to the onset of load induced packet loss or ECN CE marks. For these technologies, the discussion of queuing in Section 6.3 does not apply, but it is still necessary to confirm that the onset of losses or ECN CE marks be at an appropriate point and progressive. If the network bottleneck does not introduce significant queuing delay, modify the procedure described in Section 6.3 to start the scan at a window equal to or slightly smaller than the test_window.

Use the procedure in Section 6.3 to sweep the window across the onset of queuing and the onset of loss. The tests below all assume that the scan emulates standard additive increase and delayed ACK by incrementing the window by one packet for every 2*target_window_size packets delivered. A scan can typically be divided into three regions: below the onset of queuing, a standing queue, and at or beyond the onset of loss.

Below the onset of queuing the RTT is typically fairly constant, and the data rate varies in proportion to the window size. Once the data rate reaches the subpath IP rate, the data rate becomes fairly constant, and the RTT increases in proportion to the increase in window size. The precise transition across the start of queuing can be identified by the maximum network power, defined to be the ratio data rate over the RTT. The network power can be computed at each window size, and the window with the maximum is taken as the start of the queuing region.

If there is random background loss (e.g. bit errors, etc), precise determination of the onset of queue induced packet loss may require multiple scans. Above the onset of queuing loss, all transport protocols are expected to experience periodic losses determined by the interaction between the congestion control and AQM algorithms. For standard congestion control algorithms the periodic losses are likely to be relatively widely spaced and the details are typically dominated by the behavior of the transport protocol itself. For the stiffened transport protocols case (with non-standard, aggressive congestion control algorithms) the details of periodic losses will be dominated by how the window increase function responds to loss.

8.2.1. Congestion Avoidance

A subpath passes the congestion avoidance standing queue test if more than target_run_length packets are delivered between the onset of queuing (as determined by the window with the maximum network power as described above) and the first loss or ECN CE mark. If this test is implemented using a standard congestion control algorithm with a clamp, it can be performed in situ in the production internet as a capacity test. For an example of such a test see [Pathdiag].

For technologies that do not have conventional queues, use the test_window in place of the onset of queuing. i.e. A subpath passes the congestion avoidance standing queue test if more than target_run_length packets are delivered between start of the scan at test_window and the first loss or ECN CE mark.

8.2.2. Bufferbloat

This test confirms that there is some mechanism to limit buffer occupancy (e.g. that prevents bufferbloat). Note that this is not strictly a requirement for single stream bulk transport capacity, however if there is no mechanism to limit buffer queue occupancy then a single stream with sufficient data to deliver is likely to cause the problems described in [RFC7567], and [wikiBloat]. This may cause only minor symptoms for the dominant flow, but has the potential to make the subpath unusable for other flows and applications.

Pass if the onset of loss occurs before a standing queue has introduced more delay than than twice target_RTT, or other well defined and specified limit. Note that there is not yet a model for how much standing queue is acceptable. The factor of two chosen here reflects a rule of thumb. In conjunction with the previous test, this test implies that the first loss should occur at a queuing delay which is between one and two times the target_RTT.

Specified RTT limits that are larger than twice the target_RTT must be fully justified in the FS-TIDS.

8.2.3. Non excessive loss

This test confirms that the onset of loss is not excessive. Pass if losses are equal or less than the increase in the cross traffic plus the test stream window increase since the previous RTT. This could be restated as non-decreasing total throughput of the subpath at the onset of loss. (Note that when there is a transient drop in subpath throughput and there is not already a standing queue, a subpath that passes other queue tests in this document will have sufficient queue space to hold one full RTT worth of data).

Note that token bucket policers will not pass this test, which is as intended. TCP often stumbles badly if more than a small fraction of the packets are dropped in one RTT. Many TCP implementations will require a timeout and slowstart to recover their self clock. Even if they can recover from the massive losses the sudden change in available capacity at the bottleneck wastes serving and front path capacity until TCP can adapt to the new rate [Policing].

8.2.4. Duplex Self Interference

This engineering test confirms a bound on the interactions between the forward data path and the ACK return path when they share a half duplex link.

Some historical half duplex technologies had the property that each direction held the channel until it completely drained its queue. When a self clocked transport protocol, such as TCP, has data and ACKs passing in opposite directions through such a link, the behavior often reverts to stop-and-wait. Each additional packet added to the window raises the observed RTT by two packet times, once as the additional packet passes through the data path, and once for the additional delay incurred by the ACK waiting on the return path.

The duplex self interference test fails if the RTT rises by more than a fixed bound above the expected queuing time computed from the excess window divided by the subpath IP Capacity. This bound must be

smaller than target_RTT/2 to avoid reverting to stop and wait behavior. (e.g. Data packets and ACKs both have to be released at least twice per RTT.)

8.3. Slowstart tests

These tests mimic slowstart: data is sent at twice the effective bottleneck rate to exercise the queue at the dominant bottleneck.

8.3.1. Full Window slowstart test

This is a capacity test to confirm that slowstart is not likely to exit prematurely. Send slowstart bursts that are target_window_size total packets.

Accumulate packet transfer statistics as described in Section 7.2 to score the outcome. Pass if it is statistically significant that the observed number of good packets delivered between losses or ECN CE marks is larger than the target_run_length. Fail if it is statistically significant that the observed interval between losses or ECN CE marks is smaller than the target_run_length.

It is deemed inconclusive if the elapsed time to send the data burst is not less than half of the time to receive the ACKs. (i.e. It is acceptable to send data too fast, but sending it slower than twice the actual bottleneck rate as indicated by the ACKs is deemed inconclusive). The headway for the slowstart bursts should be the target_RTT.

Note that these are the same parameters as the Sender Full Window burst test, except the burst rate is at slowstart rate, rather than sender interface rate.

8.3.2. Slowstart AQM test

Do a continuous slowstart (send data continuously at twice the implied IP bottleneck capacity), until the first loss, stop, allow the network to drain and repeat, gathering statistics on how many packets were delivered before the loss, the pattern of losses, maximum observed RTT and window size. Justify the results. There is not currently sufficient theory justifying requiring any particular result, however design decisions that affect the outcome of this tests also affect how the network balances between long and short flows (the "mice vs elephants" problem). The queue sojourn time for the first packet delivered after the first loss should be at least one half of the target_RTT.

This is an engineering test: It should be performed on a quiescent network or testbed, since cross traffic has the potential to change the results in ill defined ways.

8.4. Sender Rate Burst tests

These tests determine how well the network can deliver bursts sent at sender's interface rate. Note that this test most heavily exercises the front path, and is likely to include infrastructure may be out of scope for an access ISP, even though the bursts might be caused by ACK compression, thinning or channel arbitration in the access ISP. See Appendix B.

Also, there are a several details about sender interface rate bursts that are not fully defined here. These details, such as the assumed sender interface rate, should be explicitly stated is a FS-TIDS.

Current standards permit TCP to send full window bursts following an application pause. (Congestion Window Validation [RFC2861] and updates to support Rate-Limited Traffic [RFC7661], are not required). Since full window bursts are consistent with standard behavior, it is desirable that the network be able to deliver such bursts, otherwise application pauses will cause unwarranted losses. Note that the AIMD sawtooth requires a peak window that is twice target_window_size, so the worst case burst may be 2*target_window_size.

It is also understood in the application and serving community that interface rate bursts have a cost to the network that has to be balanced against other costs in the servers themselves. For example TCP Segmentation Offload (TSO) reduces server CPU in exchange for larger network bursts, which increase the stress on network buffer memory. Some newer TCP implementations can pace traffic at scale [TSO_pacing][TSO_fq_pacing]. It remains to be determined if and how quickly these changes will be deployed.

There is not yet theory to unify these costs or to provide a framework for trying to optimize global efficiency. We do not yet have a model for how much server rate bursts should be tolerated by the network. Some bursts must be tolerated by the network, but it is probably unreasonable to expect the network to be able to efficiently deliver all data as a series of bursts.

For this reason, this is the only test for which we encourage derating. A TIDS could include a table of pairs of derating parameters: burst sizes and how much each burst size is permitted to reduce the run length, relative to to the target_run_length.

8.5. Combined and Implicit Tests

Combined tests efficiently confirm multiple network properties in a single test, possibly as a side effect of normal content delivery. They require less measurement traffic than other testing strategies at the cost of conflating diagnostic signatures when they fail. These are by far the most efficient for monitoring networks that are nominally expected to pass all tests.

8.5.1. Sustained Bursts Test

The sustained burst test implements a combined worst case version of all of the capacity tests above. It is simply:

Send target_window_size bursts of packets at server interface rate with target_RTT burst headway (burst start to next burst start). Verify that the observed packet transfer statistics meets the target_run_length.

Key observations:

- The subpath under test is expected to go idle for some fraction of the time, determined by the difference between the time to drain the queue at the subpath_IP_capacity, and the target_RTT. If the queue does not drain completely it may be an indication that the the subpath has insufficient IP capacity or that there is some other problem with the test (e.g. inconclusive).
- The burst sensitivity can be derated by sending smaller bursts more frequently. E.g. send target_window_size*derate packet bursts every target_RTT*derate, where "derate" is less than one.
- When not derated, this test is the most strenuous capacity test.
- A subpath that passes this test is likely to be able to sustain higher rates (close to subpath_IP_capacity) for paths with RTTs significantly smaller than the target_RTT.
- This test can be implemented with instrumented TCP [RFC4898], using a specialized measurement application at one end [MBMSource] and a minimal service at the other end [RFC0863] [RFC0864].
- This test is efficient to implement, since it does not require per-packet timers, and can make use of TSO in modern NIC hardware.
- If a subpath is known to pass the Standing Queue engineering tests (particularly that it has a progressive onset of loss at an appropriate queue depth), then the Sustained Burst Test is sufficient to assure that the subpath under test will not impair Bulk Transport Capacity at the target performance under all conditions. See Section 8.2 for a discussion of the standing queue tests.

Note that this test is clearly independent of the subpath RTT, or other details of the measurement infrastructure, as long as the measurement infrastructure can accurately and reliably deliver the required bursts to the subpath under test.

8.5.2. Passive Measurements

Any non-throughput maximizing application, such as fixed rate streaming media, can be used to implement passive or hybrid (defined in [RFC7799]) versions of Model Based Metrics with some additional instrumentation and possibly a traffic shaper or other controls in the servers. The essential requirement is that the data transmission be constrained such that even with arbitrary application pauses and bursts, the data rate and burst sizes stay within the envelope defined by the individual tests described above.

If the application's serving data rate can be constrained to be less than or equal to the target_data_rate and the serving_RTT (the RTT between the sender and client) is less than the target_RTT, this constraint is most easily implemented by clamping the transport window size to serving_window_clamp, set to the test_window, computed for the actual serving path.

Under the above constraints the serving_window_clamp will limit the both the serving data rate and burst sizes to be no larger than the procedures in Section 8.1.2 and Section 8.4 or Section 8.5.1. Since the serving RTT is smaller than the target_RTT, the worst case bursts that might be generated under these conditions will be smaller than called for by Section 8.4 and the sender rate burst sizes are implicitly derated by the serving_window_clamp divided by the target_window_size at the very least. (Depending on the application behavior, the data might be significantly smoother than specified by any of the burst tests.)

In an alternative implementation the data rate and bursts might be explicitly controlled by a programmable traffic shaper or pacing at the sender. This would provide better control over transmissions but is more complicated to implement, although the required technology is available [TSO_pacing][TSO_fq_pacing].

Note that these techniques can be applied to any content delivery that can operated at a constrained data rate to inhibit TCP equilibrium behavior.

Furthermore note that Dynamic Adaptive Streaming over HTTP (DASH) is generally in conflict with passive Model Based Metrics measurement, because it is a rate maximizing protocol. It can still meet the requirement here if the rate can be capped, for example by knowing a

priori the maximum rate needed to deliver a particular piece of content.

9. An Example

In this section we illustrate a TIDS designed to confirm that an access ISP can reliably deliver HD video from multiple content providers to all of their customers. With modern codecs, minimal HD video (720p) generally fits in 2.5 Mb/s. Due to their geographical size, network topology and modem characteristics the ISP determines that most content is within a 50 mS RTT of their users (This example RTT is a sufficient to cover the propagation delay to continental Europe or either US coast with low delay modems or somewhat smaller geographical regions if the modems require additional delay to implement advanced compression and error recovery).

2.5 Mb/s over a 50 ms path

End-to-End Parameter	value	units
target_rate	2.5	Mb/s
target_RTT	50	ms
target_MTU	1500	bytes
header_overhead	64	bytes
target_window_size	11	packets
target_run_length	363	packets

Table 1

Table 1 shows the default TCP model with no derating, and as such is quite conservative. The simplest TIDS would be to use the sustained burst test, described in Section 8.5.1. Such a test would send 11 packet bursts every 50mS, and confirming that there was no more than 1 packet loss per 33 bursts (363 total packets in 1.650 seconds).

Since this number represents is the entire end-to-end loss budget, independent subpath tests could be implemented by apportioning the packet loss ratio across subpaths. For example 50% of the losses might be allocated to the access or last mile link to the user, 40% to the network interconnections with other ISPs and 1% to each internal hop (assuming no more than 10 internal hops). Then all of the subpaths can be tested independently, and the spatial composition of passing subpaths would be expected to be within the end-to-end loss budget.

9.1. Observations about applicability

Guidance on deploying and using MBM belong in a future document. However this example illustrates some the issues that may need to be considered.

Note that another ISP, with different geographical coverage, topology or modem technology may need to assume a different target_RTT, and as a consequence different target_window_size and target_run_length, even for the same target_data rate. One of the implications of this is that infrastructure shared by multiple ISPs, such as inter-exchange points (IXPs) and other interconnects may need to be evaluated on the basis of the most stringent target_window_size and target_run_length of any participating ISP. One way to do this might be to choose target parameters for evaluating such shared infrastructure on the basis of a hypothetical reference path that does not necessarily match any actual paths.

Testing interconnects has generally been problematic: conventional performance tests run between measurement points adjacent to either side of the interconnect are not generally useful. Unconstrained TCP tests, such as iPerf [iPerf] are usually overly aggressive due to the small RTT (often less than 1 mS). With a short RTT these tools are likely to report inflated data rates because on a short RTT these tools can tolerate very high packet loss ratios and can push other cross traffic off of the network. As a consequence these measurements are useless for predicting actual user performance over longer paths, and may themselves be quite disruptive. Model Based Metrics solves this problem. The interconnect can be evaluated with the same TIDS as other subpaths. Continuing our example, if the interconnect is apportioned 40% of the losses, 11 packet bursts sent every 50mS should have fewer than one loss per 82 bursts (902 packets).

10. Validation

Since some aspects of the models are likely to be too conservative, Section 5.2 permits alternate protocol models and Section 5.3 permits test parameter derating. If either of these techniques are used, we require demonstrations that such a TIDS can robustly detect subpaths that will prevent authentic applications using state-of-the-art protocol implementations from meeting the specified Target Transport Performance. This correctness criteria is potentially difficult to prove, because it implicitly requires validating a TIDS against all possible paths and subpaths. The procedures described here are still experimental.

We suggest two approaches, both of which should be applied: first, publish a fully open description of the TIDS, including what assumptions were used and and how it was derived, such that the research community can evaluate the design decisions, test them and comment on their applicability; and second, demonstrate that applications do meet the Target Transport Performance when running over a network testbed which has the tightest possible constraints that still allow the tests in the TIDS to pass.

This procedure resembles an epsilon-delta proof in calculus. Construct a test network such that all of the individual tests of the TIDS pass by only small (infinitesimal) margins, and demonstrate that a variety of authentic applications running over real TCP implementations (or other protocols as appropriate) meets the Target Transport Performance over such a network. The workloads should include multiple types of streaming media and transaction oriented short flows (e.g. synthetic web traffic).

For example, for the HD streaming video TIDS described in Section 9, the IP capacity should be exactly the header_overhead above 2.5 Mb/s, the per packet random background loss ratio should be 1/363, for a run length of 363 packets, the bottleneck queue should be 11 packets and the front path should have just enough buffering to withstand 11 packet interface rate bursts. We want every one of the TIDS tests to fail if we slightly increase the relevant test parameter, so for example sending a 12 packet burst should cause excess (possibly deterministic) packet drops at the dominant queue at the bottleneck. This network has the tightest possible constraints that can be expected to pass the TIDS, yet it should be possible for a real application using a stock TCP implementation in the vendor's default configuration to attain 2.5 Mb/s over an 50 mS path.

The most difficult part of setting up such a testbed is arranging for it to have the tightest possible constraints that still allow it to pass the individual tests. Two approaches are suggested: constraining (configuring) the network devices not to use all available resources (e.g. by limiting available buffer space or data rate); and pre-loading subpaths with cross traffic. Note that is it important that a single tightly constrained environment just barely passes all tests, otherwise there is a chance that TCP can exploit extra latitude in some parameters (such as data rate) to partially compensate for constraints in other parameters (queue space, or vice-versa).

To the extent that a TIDS is used to inform public dialog it should be fully publicly documented, including the details of the tests, what assumptions were used and how it was derived. All of the details of the validation experiment should also be published with

sufficient detail for the experiments to be replicated by other researchers. All components should either be open source of fully described proprietary implementations that are available to the research community.

11. Security Considerations

Measurement is often used to inform business and policy decisions, and as a consequence is potentially subject to manipulation. Model Based Metrics are expected to be a huge step forward because equivalent measurements can be performed from multiple vantage points, such that performance claims can be independently validated by multiple parties.

Much of the acrimony in the Net Neutrality debate is due to the historical lack of any effective vantage independent tools to characterize network performance. Traditional methods for measuring Bulk Transport Capacity are sensitive to RTT and as a consequence often yield very different results when run local to an ISP or interconnect and when run over a customer's complete path. Neither the ISP nor customer can repeat the others measurements, leading to high levels of distrust and acrimony. Model Based Metrics are expected to greatly improve this situation.

Note that in situ measurements sometimes requires sending synthetic measurement traffic between arbitrary locations in the network, and as such are potentially attractive platforms for launching DDOS attacks. All active measurement tools and protocols must be designed to minimize the opportunities for these misuses. See the discussion in section 7 of [RFC7594].

Some of the tests described in the note are not intended for frequent network monitoring since they have the potential to cause high network loads and might adversely affect other traffic.

This document only describes a framework for designing Fully Specified Targeted IP Diagnostic Suite. Each FS-TIDS must include its own security section.

12. Acknowledgments

Ganga Maguluri suggested the statistical test for measuring loss probability in the target run length. Alex Gilgur and Merry Mou for helping with the statistics.

Meredith Whittaker for improving the clarity of the communications.

Ruediger Geib provided feedback which greatly improved the document.

This work was inspired by Measurement Lab: open tools running on an open platform, using open tools to collect open data. See http://www.measurementlab.net/

13. IANA Considerations

This document has no actions for IANA.

14. Informative References

[RFC0863] Postel, J., "Discard Protocol", STD 21, RFC 863, May 1983.

[RFC0864] Postel, J., "Character Generator Protocol", STD 22, RFC 864, May 1983.

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

[RFC2861] Handley, M., Padhye, J., and S. Floyd, "TCP Congestion Window Validation", RFC 2861, June 2000.

[RFC3148] Mathis, M. and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", RFC 3148, July 2001.

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

[RFC3465] Allman, M., "TCP Congestion Control with Appropriate Byte Counting (ABC)", RFC 3465, February 2003.

[RFC4015] Ludwig, R. and A. Gurtov, "The Eifel Response Algorithm for TCP", RFC 4015, February 2005.

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, November 2006.

[RFC4898] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, May 2007.

[RFC5136] Chimento, P. and J. Ishac, "Defining Network Capacity", RFC 5136, February 2008.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[RFC5827] Allman, M., Avrachenkov, K., Ayesta, U., Blanton, J., and P. Hurtig, "Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)", RFC 5827, DOI 10.17487/RFC5827, May 2010, <http://www.rfc-editor.org/info/rfc5827>.

[RFC5835] Morton, A. and S. Van den Berghe, "Framework for Metric Composition", RFC 5835, April 2010.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, January 2011.

[RFC6576] Geib, R., Ed., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, DOI 10.17487/RFC6576, March 2012, <http://www.rfc-editor.org/info/rfc6576>.

[RFC6673] Morton, A., "Round-Trip Packet Loss Metrics", RFC 6673, August 2012.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <http://www.rfc-editor.org/info/rfc6928>.

[RFC7312] Fabini, J. and A. Morton, "Advanced Stream and Sampling Framework for IP Performance Metrics (IPPM)", RFC 7312, August 2014.

[RFC7398] Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A Reference Path and Measurement Points for Large-Scale Measurement of Broadband Performance", RFC 7398, February 2015.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

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

[RFC7661] Fairhurst, G., Sathiaseelan, A., and R. Secchi, "Updating TCP to Support Rate-Limited Traffic", RFC 7661, DOI 10.17487/RFC7661, October 2015, <http://www.rfc-editor.org/info/rfc7661>.

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

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

[I-D.ietf-tcpm-rack]
Cheng, Y., Cardwell, N., and N. Dukkipati, "RACK: a time-based fast loss detection algorithm for TCP", draft-ietf-tcpm-rack-02 (work in progress), March 2017.

[MSMO97] Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", Computer Communications Review volume 27, number3, July 1997.

[WPING] Mathis, M., "Windowed Ping: An IP Level Performance Diagnostic", INET 94, June 1994.

[mpingSource]
Fan, X., Mathis, M., and D. Hamon, "Git Repository for mping: An IP Level Performance Diagnostic", Sept 2013, <https://github.com/m-lab/mping>.

[MBMSource]
Hamon, D., Stuart, S., and H. Chen, "Git Repository for Model Based Metrics", Sept 2013, <https://github.com/m-lab/MBM>.

[Pathdiag]
Mathis, M., Heffner, J., O'Neil, P., and P. Siemsen, "Pathdiag: Automated TCP Diagnosis", Passive and Active Measurement , June 2008.

[iPerf] Wikipedia Contributors, , "iPerf", Wikipedia, The Free Encyclopedia , cited March 2015, <http://en.wikipedia.org/w/index.php?title=Iperf&oldid=649720021>.

[Wald45] Wald, A., "Sequential Tests of Statistical Hypotheses", The Annals of Mathematical Statistics, Vol. 16, No. 2, pp. 117-186, Published by: Institute of Mathematical Statistics, Stable URL: http://www.jstor.org/stable/2235829, June 1945.

[Montgomery90] Montgomery, D., "Introduction to Statistical Quality Control - 2nd ed.", ISBN 0-471-51988-X, 1990.

[Rtool] R Development Core Team, , "R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/", , 2011.

[CVST] Krueger, T. and M. Braun, "R package: Fast Cross-Validation via Sequential Testing", version 0.1, 11 2012.

[AFD] Pan, R., Breslau, L., Prabhakar, B., and S. Shenker, "Approximate fairness through differential dropping", SIGCOMM Comput. Commun. Rev. 33, 2, April 2003.

[wikiBloat] Wikipedia, , "Bufferbloat", http://en.wikipedia.org/w/ index.php?title=Bufferbloat&oldid=608805474, March 2015.

[CCscaling] Fernando, F., Doyle, J., and S. Steven, "Scalable laws for stable network congestion control", Proceedings of Conference on Decision and Control, http://www.ee.ucla.edu/~paganini, December 2001.

[TSO_pacing] Corbet, J., "TSO sizing and the FQ scheduler", LWN.net https://lwn.net/Articles/564978/, Aug 2013.

[TSO_fq_pacing] Dumazet, E. and Y. Chen, "TSO, fair queuing, pacing: three's a charm", Proceedings of IETF 88, TCPM WG https://www.ietf.org/proceedings/88/slides/slides-88-tcpm-9.pdf, Nov 2013.

[Policing] Flach, T., Papageorge, P., Terzis, A., Pedrosa, L., Cheng, Y., Karim, T., Katz-Bassett, E., and R. Govindan, "An Internet-Wide Analysis of Traffic Policing", ACM SIGCOMM , August 2016.

Appendix A. Model Derivations

The reference target_run_length described in Section 5.2 is based on very conservative assumptions: that all excess data in flight (window) above the target_window_size contributes to a standing queue that raises the RTT, and that classic Reno congestion control with delayed ACKs are in effect. In this section we provide two alternative calculations using different assumptions.

It may seem out of place to allow such latitude in a measurement method, but this section provides offsetting requirements.

The estimates provided by these models make the most sense if network performance is viewed logarithmically. In the operational Internet, data rates span more than 8 orders of magnitude, RTT spans more than 3 orders of magnitude, and packet loss ratio spans at least 8 orders of magnitude if not more. When viewed logarithmically (as in decibels), these correspond to 80 dB of dynamic range. On an 80 dB scale, a 3 dB error is less than 4% of the scale, even though it represents a factor of 2 in untransformed parameter.

This document gives a lot of latitude for calculating target_run_length, however people designing a TIDS should consider the effect of their choices on the ongoing tussle about the relevance of "TCP friendliness" as an appropriate model for Internet capacity allocation. Choosing a target_run_length that is substantially smaller than the reference target_run_length specified in Section 5.2 strengthens the argument that it may be appropriate to abandon "TCP friendliness" as the Internet fairness model. This gives developers incentive and permission to develop even more aggressive applications and protocols, for example by increasing the number of connections that they open concurrently.

A.1. Queueless Reno

In Section 5.2 models were derived based on the assumption that the subpath IP rate matches the target rate plus overhead, such that the excess window needed for the AIMD sawtooth causes a fluctuating queue at the bottleneck.

An alternate situation would be a bottleneck where there is no significant queue and losses are caused by some mechanism that does not involve extra delay, for example by the use of a virtual queue as done in Approximate Fair Dropping [AFD]. A flow controlled by such a bottleneck would have a constant RTT and a data rate that fluctuates in a sawtooth due to AIMD congestion control. Assume the losses are being controlled to make the average data rate meet some goal which

is equal or greater than the target_rate. The necessary run length to meet the target_rate can be computed as follows:

For some value of Wmin, the window will sweep from Wmin packets to 2*Wmin packets in 2*Wmin RTT (due to delayed ACK). Unlike the queuing case where Wmin = target_window_size, we want the average of Wmin and 2*Wmin to be the target_window_size, so the average data rate is the target rate. Thus we want Wmin = (2/3)*target_window_size.

Between losses each sawtooth delivers (1/2)(Wmin+2*Wmin)(2Wmin) packets in 2*Wmin round trip times.

Substituting these together we get:

target_run_length = (4/3)(target_window_size^2)

Note that this is 44% of the reference_run_length computed earlier. This makes sense because under the assumptions in Section 5.2 the AMID sawtooth caused a queue at the bottleneck, which raised the effective RTT by 50%.

Appendix B. The effects of ACK scheduling

For many network technologies simple queuing models don’t apply: the network schedules, thins or otherwise alters the timing of ACKs and data, generally to raise the efficiency of the channel allocation algorithms when confronted with relatively widely spaced small ACKs. These efficiency strategies are ubiquitous for half duplex, wireless and broadcast media.

Altering the ACK stream by holding or thinning ACKs typically has two consequences: it raises the implied bottleneck IP capacity, making the fine grained slowstart bursts either faster or larger and it raises the effective RTT by the average time that the ACKs and data are delayed. The first effect can be partially mitigated by re-clocking ACKs once they are beyond the bottleneck on the return path to the sender, however this further raises the effective RTT.

The most extreme example of this sort of behavior would be a half duplex channel that is not released as long as the endpoint currently holding the channel has more traffic (data or ACKs) to send. Such environments cause self clocked protocols under full load to revert to extremely inefficient stop and wait behavior. The channel constrains the protocol to send an entire window of data as a single contiguous burst on the forward path, followed by the entire window of ACKs on the return path.

If a particular return path contains a subpath or device that alters the timing of the ACK stream, then the entire front path from the sender up to the bottleneck must be tested at the burst parameters implied by the ACK scheduling algorithm. The most important parameter is the Implied Bottleneck IP Capacity, which is the average rate at which the ACKs advance snd.una. Note that thinning the ACK stream (relying on the cumulative nature of seg.ack to permit discarding some ACKs) causes most TCP implementations to send interface rate bursts to offset the longer times between ACKs in order to maintain the average data rate.

Note that due to ubiquitous self clocking in Internet protocols, ill conceived channel allocation mechanisms are likely to increases the queuing stress on the front path because they cause larger full sender rate data bursts.

Holding data or ACKs for channel allocation or other reasons (such as forward error correction) always raises the effective RTT relative to the minimum delay for the path. Therefore it may be necessary to replace target_RTT in the calculation in Section 5.2 by an effective_RTT, which includes the target_RTT plus a term to account for the extra delays introduced by these mechanisms.

Appendix C. Version Control

This section to be removed prior to publication.

Formatted: Thu Apr 7 18:12:37 PDT 2016

Authors' Addresses

Matt Mathis
Google, Inc
1600 Amphitheater Parkway
Mountain View, California 94043
USA

Email: mattmathis@google.com

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

Phone: +1 732 420 1571
Email: acmorton@att.com

IPPM WG
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

R. Civil
Ciena Corporation
A. Morton
AT&T Labs
R. Rahman
Cisco Systems
M. Jethanandani
Xoriant Corporation
K. Pentikousis, Ed.
Travelping
July 2, 2018

Two-Way Active Measurement Protocol (TWAMP) Data Model
draft-ietf-ippm-twamp-yang-13

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

Table of Contents

1. Introduction

The Two-Way Active Measurement Protocol (TWAMP) [RFC5357] is used to measure network performance parameters such as latency, bandwidth, and packet loss by sending probe packets and measuring their experience in the network. To date, TWAMP implementations do not come with a standard management framework, and, as such, implementers have no choice except to provide a proprietary mechanism. This document addresses this gap by defining the model using UML [UML] class diagrams, and formally specifying a NMDA-complaint [RFC8342] TWAMP data model using YANG 1.1 [RFC7950].

1.1. Motivation

In current TWAMP deployments the lack of a standardized data model limits the flexibility to dynamically instantiate TWAMP-based measurements across equipment from different vendors. In large, virtualized, and dynamically instantiated infrastructures where network functions are placed according to orchestration algorithms, proprietary mechanisms for managing TWAMP measurements pose severe limitations with respect to programmability.

Two major trends call for standardizing TWAMP management aspects. First, it is expected that in the coming years large-scale and multi-vendor TWAMP deployments will become the norm. From an operations perspective, using several vendor-specific TWAMP configuration mechanisms when one standard mechanism could provide an alternative is expensive and inefficient. Second, the increasingly software-defined and virtualized nature of network infrastructures, based on dynamic service chains [NSC] and programmable control and management planes Software-Defined Networking (SDN): Layers and Architecture Terminology [RFC7426] requires a well-defined data model for TWAMP implementations. This document defines such a TWAMP data model and specifies it formally using the YANG 1.1 [RFC7950] data modeling language.

Note to RFC Editor:

Please replace the date 2018-07-02 in Section 5.2 of the draft with the date of publication of this draft as a RFC. Also, replace reference to RFC XXXX, and draft-ietf-ippm-port-twamp-test with the RFC numbers assigned to the drafts.

1.2. Terminology

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

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Document Organization

The rest of this document is organized as follows. Section 2 presents the scope and applicability of this document. Section 3 provides a high-level overview of the TWAMP data model. Section 4 details the configuration parameters of the data model and Section 5 specifies in YANG the TWAMP data model. Section 6 lists illustrative examples which conform to the YANG data model specified in this document. Appendix A elaborates these examples further.

2. Scope, Model, and Applicability

The purpose of this document is the specification of a vendor-independent data model for TWAMP implementations.

Figure 1 illustrates a redrawn version of the TWAMP logical model found in Section 1.2 of TWAMP [RFC5357]. The figure is annotated with pointers to the UML [UML] diagrams provided in this document and associated with the data model of the four logical entities in a TWAMP deployment, namely the TWAMP Control-Client, Server, Session-Sender and Session-Reflector. A UML [UML] Notation Guide is available in Section 5 of the said document.

As per TWAMP [RFC5357], unlabeled links in Figure 1 are left unspecified and may be proprietary protocols.

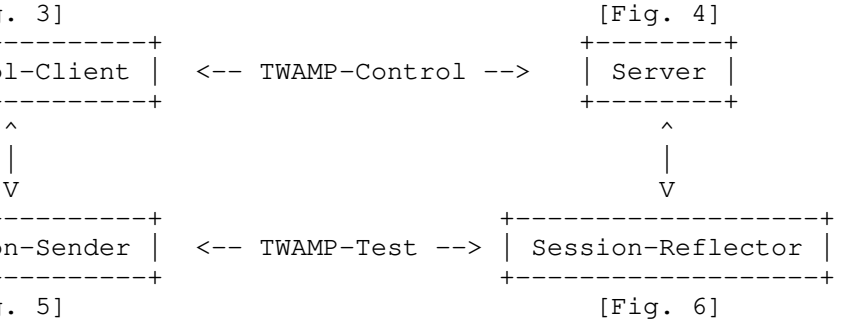


Figure 1: Annotated TWAMP logical model

As per TWAMP [RFC5357], a TWAMP implementation may follow a simplified logical model, in which the same node acts both as Control-Client and Session-Sender, while another node acts at the same time as TWAMP Server and Session-Reflector. Figure 2 illustrates this simplified logical model and indicates the

interaction between the TWAMP configuration client and server using, for instance, NETCONF [RFC6241] or RESTCONF [RFC8040].

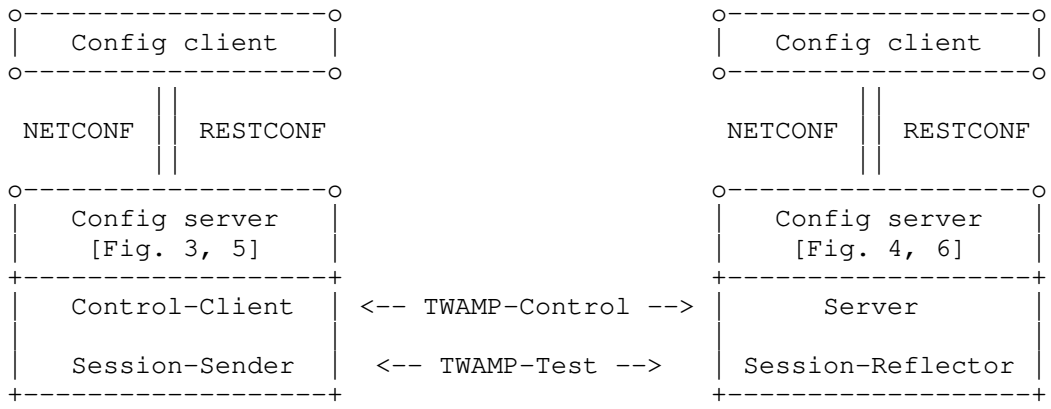


Figure 2: Simplified TWAMP model and protocols

The data model defined in this document is orthogonal to the specific protocol used between the Config client and Config server to communicate the TWAMP configuration parameters.

Operational actions such as how TWAMP-Test sessions are started and stopped, how performance measurement results are retrieved, or how stored results are cleared, and so on, are not addressed by the configuration model defined in this document. As noted above, such operational actions are not part of the TWAMP specification TWAMP [RFC5357] and hence are out of scope of this document. See also Appendix B. In addition, for operational state, current work in Registry for Performance Metrics [I-D.ietf-ippm-metric-registry], can be used to develop an independent model for the performance metrics that need to be captured and retrieved.

3. Data Model Overview

The TWAMP data model includes four categories of configuration items.

First, global configuration items relate to parameters that are set on a per device level. For example, the administrative status of the device with respect to whether it allows TWAMP sessions and, if so, in what capacity (e.g. Control-Client, Server or both), is a typical instance of a global configuration item.

A second category includes attributes that can be configured on a per TWAMP-Control connection basis, such as the Server IP address.

A third category includes attributes related to per TWAMP-Test session attributes, for instance setting different values in the Differentiated Services Code Point (DSCP) field.

Finally, the data model includes attributes that relate to the operational state of the TWAMP implementation.

As the TWAMP data model is described in the remaining sections of this document, readers should keep in mind the functional entity grouping illustrated in Figure 1.

3.1. Control-Client

A TWAMP Control-Client has an administrative status field set at the device level that indicates whether the node is enabled to function as such.

Each TWAMP Control-Client is associated with zero or more TWAMP-Control connections. The main configuration parameters of each control connection are:

- 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 TWAMP Server.
- Authentication and encryption attributes such as KeyID, Token and the Client Initialization Vector (Client-IV); see also Section 3.1 in OWAMP [RFC4656] and Randomness Requirements for Security [RFC4086].

Each TWAMP-Control connection, in turn, is associated with zero or more TWAMP-Test sessions. For each test session, the following configuration items should be noted:

- The test session name uniquely identifies a particular test session at the Control-Client and Session-Sender. Similar to the control connections above, this unique test session name is needed because at the time of creation of a TWAMP-Test session, for example, the source UDP port number is not known to uniquely identify the test session.

- The IP address and UDP port number of the Session-Sender on the path under test by TWAMP.

- The IP address and UDP port number of the Session-Reflector on said path.

- Information pertaining to the test packet stream, such as the test starting time, which performance metric is to be used, as defined in Registry for Performance Metrics [I-D.ietf-ippm-metric-registry], or whether the test should be repeated.

3.2. Server

Each TWAMP Server has an administrative status field set at the device level to indicate whether the node is enabled to function as a TWAMP Server.

Each Server is associated with zero or more TWAMP-Control connections. Each control connection is uniquely identified by the 4-tuple {Control-Client IP address, Control-Client TCP port number, Server IP address, Server TCP port}. Control connection configuration items on a TWAMP Server are read-only.

3.3. Session-Sender

A TWAMP Session-Sender has an administrative status field set at the device level that indicates whether the node is enabled to function as such.

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

- The test session name MUST be identical to 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, the number of test packets and the packet distribution to be employed; see also Network performance measurement with periodic streams [RFC3432].

3.4. Session-Reflector

Each TWAMP Session-Reflector has an administrative status field set at the device level to indicate whether the node is enabled to function as such.

Each Session-Reflector is associated with zero or more TWAMP-Test sessions. For each test session, the REFWAIT timeout parameter, which determines whether to discontinue the session if no packets have been received (TWAMP [RFC5357], Section 4.2), can be configured.

Read-only access to other data model parameters, such as the Sender IP address, is foreseen. Each test session can be uniquely identified by the 4-tuple mentioned in Section 3.2.

4. Data Model Parameters

This section defines the TWAMP data model using UML [UML] and introduces selected parameters associated with the four TWAMP logical entities. The complete TWAMP data model specification is provided in the YANG module presented in Section 5.2.

4.1. Control-Client

The client container (see Figure 3) holds items that are related to the configuration of the TWAMP Control-Client logical entity (recall Figure 1).

The client container includes an administrative configuration parameter (client/admin-state) that indicates whether the device is allowed to initiate TWAMP-Control connections.

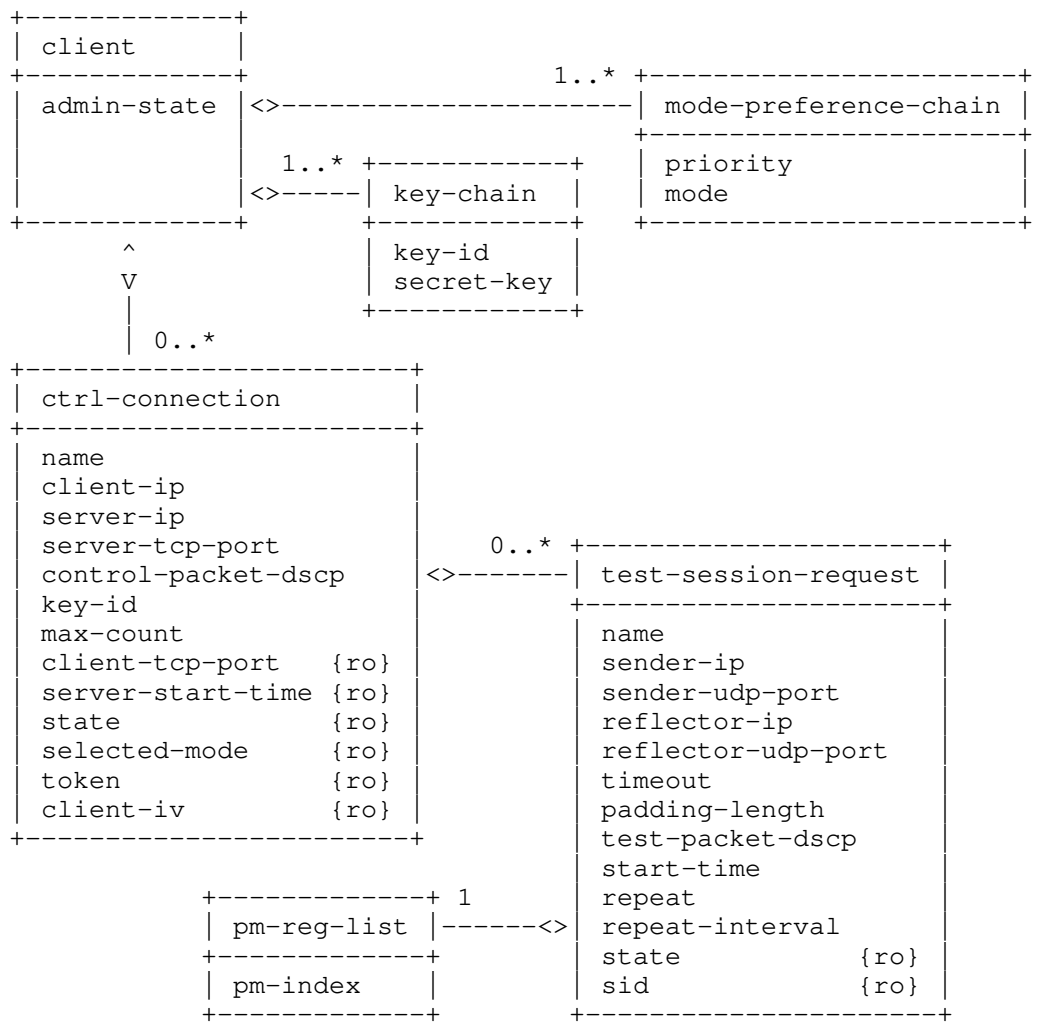


Figure 3: TWAMP Control-Client UML class diagram

The client container holds a list (mode-preference-chain) which specifies the Mode values according to their preferred order of use by the operator of this Control-Client, including the authentication and encryption Modes. Specifically, mode-preference-chain lists the mode and its corresponding priority, as a 16-bit unsigned integer. Values for the priority start with zero, the highest priority, and decreasing priority value is indicated by every increase in value by one.

Depending on the Modes available in the Server Greeting, the Control-Client MUST choose the highest priority Mode from the configured mode-preference-chain list.

Note that the list of preferred Modes may set multiple bit positions independently, such as when referring to the extended TWAMP features in Mixed Security Mode for TWAMP [RFC5618], Individual Session Control Feature for TWAMP [RFC5938], TWAMP Reflect Octets and Symmetrical Size Features [RFC6038], and IKEv2-Derived Shared Secret Key for OWAMP and TWAMP [RFC7717]. If the Control-Client cannot determine an acceptable Mode, or when the bit combinations do not make sense, e.g., both authenticated and unauthenticated bit are set, it MUST respond with zero Mode bits set in the Set-up Response message, indicating it will not continue with the control connection.

In addition, the client container holds a list named key-chain which relates key-id with the respective secret-key. Both the Server and the Control-Client use the same mappings from key-id to secret-key (in Figure 3); in order for this to work properly, key-id must be unique across all systems in the administrative domain. The Server, being prepared to conduct sessions with more than one Control-Client, uses key-id to choose the appropriate secret-key; a Control-Client would typically have different secret keys for different Servers. The secret-key is the shared secret, of type binary and the length SHOULD contain at least 128 bits of entropy. The key-id and secret-key encoding SHOULD follow Section 9.8 of YANG [RFC7950]. The derived key length (dkLen in PKCS #5: Password-Based Cryptography Specification Version 2.1 [RFC8018]) MUST be 16 octets for the AES Session-key used for encryption and 32 octets for the HMAC-SHA1 Session-key used for authentication; see also Section 6.10 of OWAMP [RFC4656].

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

In turn, each ctrl-connection holds a test-session-request list. Each test-session-request holds information associated with the Control-Client for this test session. This includes information associated with the Request-TW-Session/Accept-Session message exchange (see Section 3.5 of TWAMP [RFC5357]).

There SHALL be one instance of test-session-request for each TWAMP-Test session that is to be negotiated by this TWAMP-Control connection via a Request-TW-Session/Accept-Session exchange.

The Control-Client is also responsible for scheduling TWAMP-Test sessions, therefore test-session-request holds information related to these actions (e.g. pm-index, repeat-interval).

4.2. Server

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

The server container includes an administrative configuration parameter (server/admin-state) that indicates whether the device is allowed to receive TWAMP-Control connections.

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

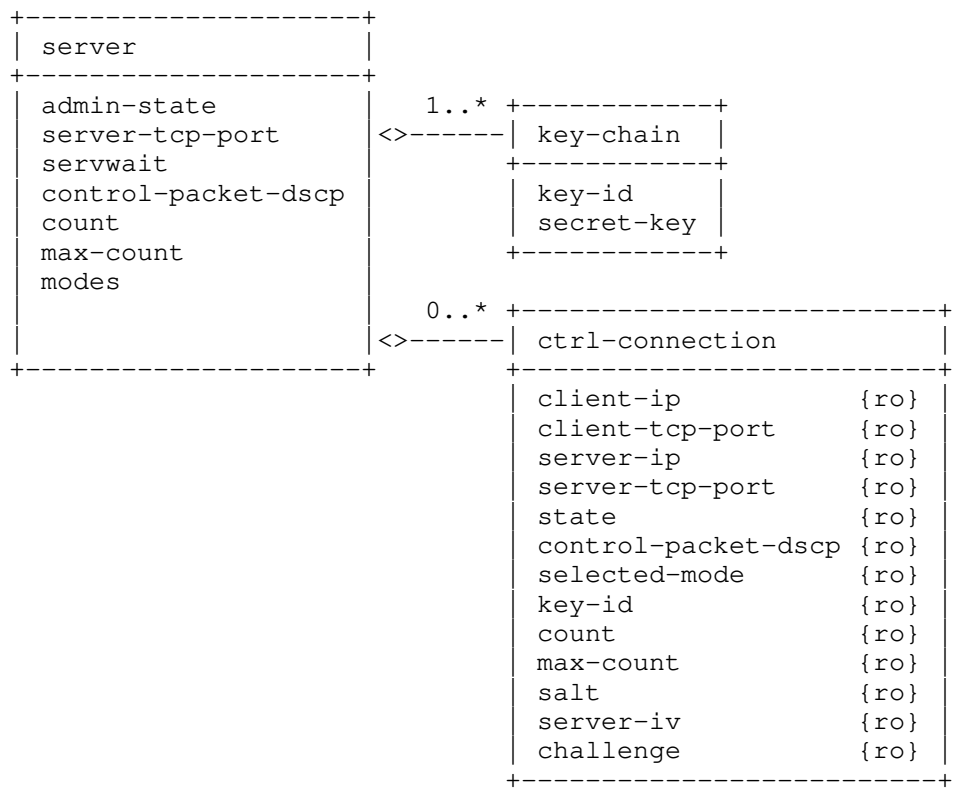


Figure 4: TWAMP Server UML class diagram

Each server container holds a list named key-chain which relates key-id with the respective secret-key. As mentioned in Section 4.1, both the Server and the Control-Client use the same mapping from key-id to shared secret-key; in order for this to work properly, key-id must be unique across all the systems in the administrative domain. The Server, being prepared to conduct sessions with more than one Control-Client, uses key-id to choose the appropriate secret-key; a Control-Client would typically have different secret keys for different Servers. The key-id tells the Server which shared secret-key the Control-Client wishes to use for authentication or encryption.

Each incoming control connection active on the Server is represented by a ctrl-connection. There SHALL be one ctrl-connection per incoming TWAMP-Control (TCP) connection that is received and active on the Server. Each ctrl-connection can be uniquely identified by the 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port}. All items in the ctrl-connection list are read-only.

4.3. Session-Sender

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

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

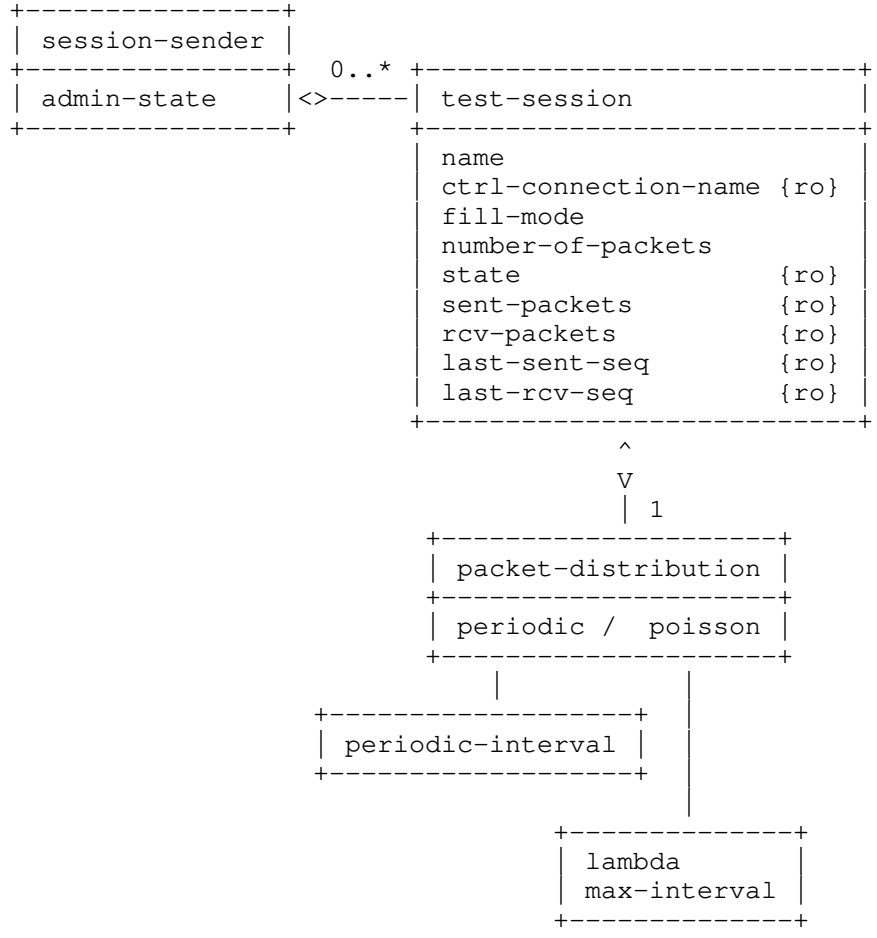


Figure 5: TWAMP Session-Sender UML class diagram

Each TWAMP-Test session initiated by the Session-Sender will be represented by an instance of a test-session object. There SHALL be

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

4.4. Session-Reflector

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

The session-reflector container includes an administrative parameter (session-reflector/admin-state) that controls whether the device is allowed to respond to incoming TWAMP-Test sessions.

A device operating in the Session-Reflector role cannot configure attributes on a per-session basis, as it has no foreknowledge of what incoming sessions it will receive. As such, any parameter that the Session-Reflector might want to apply to an incoming TWAMP-Test session must be configured at the overall Session-Reflector level and are applied to all incoming sessions.

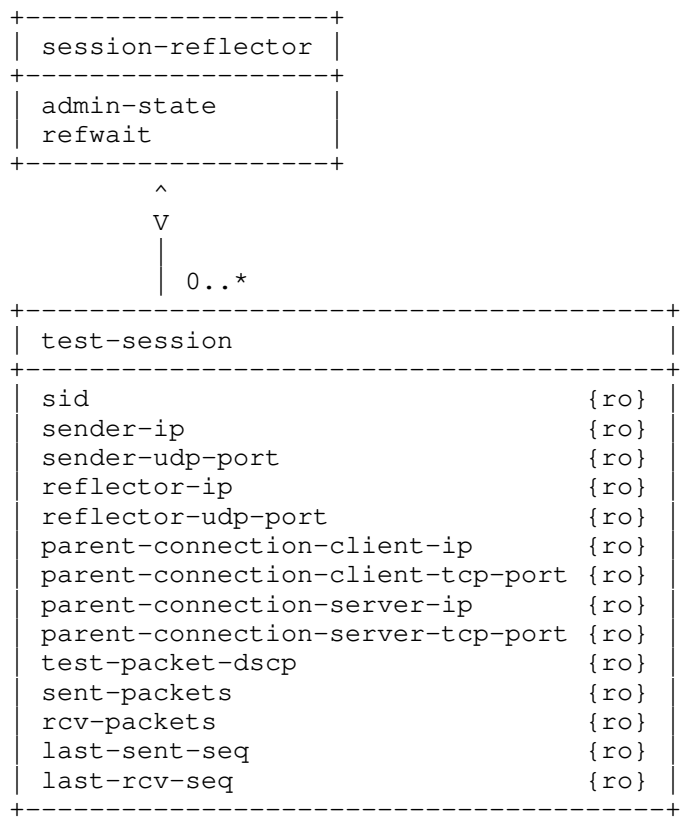


Figure 6: TWAMP Session-Reflector UML class diagram

Each incoming TWAMP-Test session that is active on the Session-Reflector SHALL be represented by an instance of a test-session object. All items in the test-session object are read-only.

Instances of test-session are indexed by a session identifier (sid). This value is auto-allocated by the TWAMP Server as test session requests are received, and communicated back to the Control-Client in the SID field of the Accept-Session message; see Section 4.3 of TWAMP Reflect Octets and Symmetrical Size Features [RFC6038].

When attempting to retrieve operational data for active test sessions from a Session-Reflector device, the user will not know what sessions are currently active on that device, or what SIDs have been auto-allocated for these test sessions. If the user has network access to the Control-Client device, then it is possible to read the data for this session under client/ctrl-connection/test-session-request/sid and obtain the SID (see Figure 3). The user may then use this SID

value as an index to retrieve an individual session-reflector/test-session instance on the Session-Reflector device.

If the user has no network access to the Control-Client device, then the only option is to retrieve all test-session instances from the Session-Reflector device, and then pick out specific test-session instances of interest to the user. This could be problematic if a large number of test sessions are currently active on that device.

Each Session-Reflector TWAMP-Test session contains the following 4-tuple: {parent-connection-client-ip, parent-connection-client-tcp-port, parent-connection-server-ip, parent-connection-server-tcp-port}. This 4-tuple MUST correspond to the equivalent 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port} in server/ctrl-connection. This 4-tuple allows the user to trace back from the TWAMP-Test session to the (parent) TWAMP-Control connection that negotiated this test session.

5. Data Model

This section formally specifies the TWAMP data model using YANG.

5.1. YANG Tree Diagram

This section presents a simplified graphical representation of the TWAMP data model using a YANG tree diagram. Readers should keep in mind that the limit of 72 characters per line forces us to introduce artificial line breaks in some tree diagram nodes. Tree diagrams used in this document follow the notation defined in YANG Tree Diagrams [RFC8340].

```
module: ietf-twamp
    +--rw twamp
       +--rw client {control-client}?
       |  +--rw admin-state?             boolean
       |  +--rw mode-preference-chain* [priority]
       |  |  +--rw priority    uint16
       |  |  +--rw mode?       twamp-modes
       |  +--rw key-chain* [key-id]
       |  |  +--rw key-id        string
       |  |  +--rw secret-key?   binary
       |  +--rw ctrl-connection* [name]
       |     +--rw name                    string
       |     +--rw client-ip?              inet:ip-address
       |     +--rw server-ip               inet:ip-address
       |     +--rw server-tcp-port?        inet:port-number
       |     +--rw control-packet-dscp?    inet:dscp
       |     +--rw key-id?                 string
```

```
|      +--rw max-count-exponent?     uint8
|      +--ro client-tcp-port?        inet:port-number
|      +--ro server-start-time?      uint64
|      +--ro repeat-count?           uint64
|      +--ro state?
|      |       control-client-connection-state
|      +--ro selected-mode?          twamp-modes
|      +--ro token?                  binary
|      +--ro client-iv?              binary
|      +--rw test-session-request* [name]
|         +--rw name                  string
|         +--rw sender-ip?            inet:ip-address
|         +--rw sender-udp-port?      union
|         +--rw reflector-ip          inet:ip-address
|         +--rw reflector-udp-port?   inet:port-number
|         +--rw timeout?              uint64
|         +--rw padding-length?       uint32
|         +--rw test-packet-dscp?     inet:dscp
|         +--rw start-time?           uint64
|         +--rw repeat?               uint32
|         +--rw repeat-interval?      uint32
|         +--rw pm-reg-list* [pm-index]
|         |  +--rw pm-index    uint16
|         +--ro state?                test-session-state
|         +--ro sid?                  string
|
+--rw server {server}?
|  +--rw admin-state?           boolean
|  +--rw server-tcp-port?       inet:port-number
|  +--rw servwait?              uint32
|  +--rw control-packet-dscp?   inet:dscp
|  +--rw count?                 uint8
|  +--rw max-count-exponent?    uint8
|  +--rw modes?                 twamp-modes
|  +--rw key-chain* [key-id]
|  |  +--rw key-id        string
|  |  +--rw secret-key?   binary
|  +--ro ctrl-connection*
|           [client-ip client-tcp-port server-ip server-tcp-port]
|     +--ro client-ip              inet:ip-address
|     +--ro client-tcp-port        inet:port-number
|     +--ro server-ip              inet:ip-address
|     +--ro server-tcp-port        inet:port-number
|     +--ro state?                 server-ctrl-connection-state
|     +--ro control-packet-dscp?   inet:dscp
|     +--ro selected-mode?         twamp-modes
|     +--ro key-id?                string
|     +--ro count?                 uint8
|     +--ro max-count-exponent?    uint8
|
```

```
        |      +--ro salt?                  binary
        |      +--ro server-iv?             binary
        |      +--ro challenge?             binary
        +--rw session-sender {session-sender}?
        |  +--rw admin-state?    boolean
        |  +--rw test-session* [name]
        |     +--rw name                    string
        |     +--ro ctrl-connection-name?   string
        |     +--rw fill-mode?              padding-fill-mode
        |     +--rw number-of-packets       uint32
        |     +--rw (packet-distribution)?
        |     |  +--:(periodic)
        |     |  |  +--rw periodic-interval       decimal64
        |     |  +--:(poisson)
        |     |     +--rw lambda                 decimal64
        |     |     +--rw max-interval?          decimal64
        |     +--ro state?                  sender-session-state
        |     +--ro sent-packets?           uint32
        |     +--ro rcv-packets?            uint32
        |     +--ro last-sent-seq?          uint32
        |     +--ro last-rcv-seq?           uint32
        +--rw session-reflector {session-reflector}?
           +--rw admin-state?    boolean
           +--rw refwait?        uint32
           +--ro test-session*
                   [sender-ip sender-udp-port reflector-ip reflector-udp
-port]
              +--ro sid?                                  string
              +--ro sender-ip                             inet:ip-address
              +--ro sender-udp-port
              |       dynamic-port-number
              +--ro reflector-ip                          inet:ip-address
              +--ro reflector-udp-port                    inet:port-numbe
r
              +--ro parent-connection-client-ip?          inet:ip-address
              +--ro parent-connection-client-tcp-port?    inet:port-numbe
r
              +--ro parent-connection-server-ip?          inet:ip-address
              +--ro parent-connection-server-tcp-port?    inet:port-numbe
r
              +--ro test-packet-dscp?                     inet:dscp
              +--ro sent-packets?                         uint32
              +--ro rcv-packets?                          uint32
              +--ro last-sent-seq?                        uint32
              +--ro last-rcv-seq?                         uint32
```

Figure 7: YANG Tree Diagram.

5.2. YANG Module

This section presents the YANG module for the TWAMP data model defined in this document. The module imports definitions from Common YANG Data Types [RFC6991], and references NTPv4 Specification [RFC5905], Framework for IP Performance Metrics [RFC2330], Randomness Requirements for Security [RFC4086], OWAMP [RFC4656], TWAMP [RFC5357], More Features for TWAMP [RFC5618], Individual Session Control Feature [RFC5938], TWAMP Reflect Octets and Symmetrical Size Features [RFC6038], Advances Stream and Sampling Framework [RFC7312], IKEv2-Derived Shared Secret Key for OWAMP and TWAMP [RFC7717], and OWAMP and TWAMP Well-Known Port Assignments [I-D.ietf-ippm-port-twamp-test].

<CODE BEGINS> file "ietf-twamp@2018-07-02.yang"

```
module ietf-twamp {
  yang-version 1.1;
  namespace urn:ietf:params:xml:ns:yang:ietf-twamp;
  prefix ietf-twamp;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Types.";
  }

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

  contact
    "WG Web: http://tools.ietf.org/wg/ippm/
     WG List: ippm@ietf.org

     Editor: Ruth Civil
             gcivil@ciena.com
     Editor: Al Morton
             acmorton@att.com
     Editor: Reshad Rehman
             rrahman@cisco.com
     Editor: Mahesh Jethanandani
             mjethanandani@gmail.com
     Editor: Kostas Pentikousis
             k.pentikousis@travelping.com";


  description
    "This YANG module specifies a vendor-independent data
```

```
        model for the Two-Way Active Measurement Protocol (TWAMP).

        The data model covers four TWAMP logical entities, namely,
        Control-Client, Server, Session-Sender, and Session-Reflector,
        as illustrated in the annotated TWAMP logical model (Fig. 1
        of RFC XXXX).

        This YANG module uses features to indicate which of the four
        logical entities are supported by a TWAMP implementation.

        Copyright (c) 2018 IETF Trust and the persons identified as
        the document authors.  All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD
        License set forth in Section 4.c of the IETF Trust's Legal
        Provisions Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

      revision 2018-07-02 {
        description
          "Initial Revision.

          Covers RFC 5357, RFC 5618, RFC 5938, RFC 6038, RFC 7717, and
          draft-ietf-ippm-metric-registry";

        reference
          "RFC XXXX: TWAMP YANG Data Model.";
      }


      /*
       * Typedefs
       */

      typedef twamp-modes {
        type bits {
          bit unauthenticated {
            position 0;
            description
              "Unauthenticated mode, in which no encryption or
              authentication is applied in TWAMP-Control and
              TWAMP-Test. KeyID, Token, and Client-IV are not used in
              the Set-Up-Response message. See Section 3.1 of
              RFC 4656.";
```

```
          reference
            "RFC 4656: A One-way Active Measurement Protocol
             (OWAMP)";
        }
        bit authenticated {
          position 1;
          description
            "Authenticated mode, in which the Control-Client and
             Server possess a shared secret thus prohibiting
             'theft of service'. As per Section 6 of RFC 4656,
             in 'authenticated mode, the timestamp is in the clear
             and is not protected cryptographically in any way,
             while the rest of the message has the same protection
             as in encrypted mode. This mode allows one to trade off
             cryptographic protection against accuracy of
             timestamps.'";
          reference
            "RFC 4656: A One-way Active Measurement Protocol
             (OWAMP)";
        }
        bit encrypted {
          position 2;
          description
            "Encrypted mode 'makes it impossible to alter
             timestamps undetectably' [Section 6 of RFC 4656].
             See also Section 4 of RFC 7717.";
          reference
            "RFC 4656: A One-way Active Measurement Protocol
             (OWAMP)";
        }
        bit unauth-test-encrpyt-control {
          position 3;
          description
            "When using the Mixed Security Mode, the TWAMP-Test
             protocol follows the Unauthenticated mode and the
             TWAMP-Control protocol the Encrypted mode.";
          reference
            "RFC 5618: Mixed Security Mode for the Two-Way Active
             Measurement Protocol (TWAMP)";
        }
        bit individual-session-control {
          position 4;
          description
            "This mode enables individual test sessions using
             Session Identifiers.";
          reference
            "RFC 5938: Individual Session Control Feature
             for the Two-Way Active Measurement Protocol (TWAMP)";
```

```
        }
        bit reflect-octets {
          position 5;
          description
            "This mode indicates the reflect octets capability.";
          reference
            "RFC 6038: Two-Way Active Measurement Protocol (TWAMP)
             Reflect Octets and Symmetrical Size Features";
        }
        bit symmetrical-size {
          position 6;
          description
            "This mode indicates support for the symmetrical size
             sender test packet format.";
          reference
            "RFC 6038: Two-Way Active Measurement Protocol (TWAMP)
             Reflect Octets and Symmetrical Size Features";
        }
        bit IKEv2Derived {
          position 7;
          description
            "In this mode the the shared key is derived
             from an IKEv2 security association (SA).";
          reference
            "RFC 7717: IKEv2-Derived Shared Secret Key for
             the One-Way Active Measurement Protocol (OWAMP)
             and Two-Way Active Measurement Protocol (TWAMP)";
        }
      }
      description
        "Specifies the configurable TWAMP-Modes supported during a
         TWAMP-Control Connection setup between a Control-Client
         and a Server. Section 7 of RFC 7717 summarizes the
         TWAMP-Modes registry and points to their formal
         specification.";
    }

    typedef control-client-connection-state {
      type enumeration {
        enum active {
          description
            "Indicates an active TWAMP-Control connection to
             Server.";
        }
        enum idle {
          description
            "Indicates an idle TWAMP-Control connection to Server.";
        }
```

```
      }
      description
        "Indicates the Control-Client TWAMP-Control connection
         state.";
    }

    typedef test-session-state {
      type enumeration {
        enum accepted {
          value 0;
          description
            "Indicates an accepted TWAMP-Test session request.";
        }
        enum failed {
          value 1;
          description
            "Indicates a TWAMP-Test session failure due to
             some unspecified reason (catch-all).";
        }
        enum internal-error {
          value 2;
          description
            "Indicates a TWAMP-Test session failure due to
             an internal error.";
        }
        enum not-supported {
          value 3;
          description
            "Indicates a TWAMP-Test session failure because
             some aspect of the TWAMP-Test session request
             is not supported.";
        }
        enum permanent-resource-limit {
          value 4;
          description
            "Indicates a TWAMP-Test session failure due to
             permanent resource limitations.";
        }
        enum temp-resource-limit {
          value 5;
          description
            "Indicates a TWAMP-Test session failure due to
             temporary resource limitations.";
        }
      }
      description
        "Indicates the Control-Client TWAMP-Test session state.";
    }
```

```
    typedef server-ctrl-connection-state {
      type enumeration {
        enum active {
          description
            "Indicates an active TWAMP-Control connection
             to the Control-Client.";
        }
        enum servwait {
          description
            "Indicates that the TWAMP-Control connection to the
             Control-Client is in SERVWAIT as per the definition of
             Section 3.1 of RFC 5357.";
        }
      }
      description
        "Indicates the Server TWAMP-Control connection state.";
    }

    typedef sender-session-state {
      type enumeration {
        enum active {
          description
            "Indicates that the TWAMP-Test session is active.";
        }
        enum failure {
          description
            "Indicates that the TWAMP-Test session has failed.";
        }
      }
      description
        "Indicates the Session-Sender TWAMP-Test session state.";
    }

    typedef padding-fill-mode {
      type enumeration {
        enum zero {
          description
            "TWAMP-Test packets are padded with all zeros.";
        }
        enum random {
          description
            "TWAMP-Test packets are padded with pseudo-random
            numbers.";
        }
      }
      description
        "Indicates what type of packet padding is used in the
         TWAMP-Test packets.";
```

```
  }

  typedef dynamic-port-number {
    type inet:port-number {
      range 49152..65535;
    }
    description "Dynamic range for port numbers.";
  }

  /*
   * Features
   */

  feature control-client {
    description
      "Indicates that the device supports configuration of the
       TWAMP Control-Client logical entity.";
  }

  feature server {
    description
      "Indicates that the device supports configuration of the
       TWAMP Server logical entity.";
  }

  feature session-sender {
    description
      "Indicates that the device supports configuration of the
       TWAMP Session-Sender logical entity.";
  }

  feature session-reflector {
    description
      "Indicates that the device supports configuration of the
       TWAMP Session-Reflector logical entity.";
  }


  /*
   * Reusable node groups
   */

  grouping key-management {
    list key-chain {
      key key-id;
      leaf key-id {
        type string {
          length 1..80;
```

```
          }
          description
            "KeyID used for a TWAMP-Control connection. As per
             Section 3.1 of RFC 4656, KeyID is 'a UTF-8 string, up to
             80 octets in length' and is used to select which 'shared
             shared secret the [Control-Client] wishes to use to
             authenticate or encrypt'.";
          }
          leaf secret-key {
            type binary;
            description
              "The secret key corresponding to the KeyID for this
               TWAMP-Control connection.";
          }
          description
            "Relates KeyIDs with their respective secret keys
             in a TWAMP-Control connection.";
      }
      description
          "Used by the Control-Client and Server for TWAMP-Control
           key management.";
    }

    grouping maintenance-statistics {
      leaf sent-packets {
        type uint32;
        config false;
        description
          "Indicates the number of packets sent.";
      }

      leaf rcv-packets {
        type uint32;
        config false;
        description
          "Indicates the number of packets received.";
      }

      leaf last-sent-seq {
        type uint32;
        config false;
        description
          "Indicates the last sent sequence number.";
      }

      leaf last-rcv-seq {
        type uint32;
        config false;
```

```
        description
          "Indicates the last received sequence number.";
      }
      description
        "Used for TWAMP-Test maintenance statistics.";
    }

    grouping count {
      leaf count {
        type uint8 {
          range "10..31";
        }
        default 15;
        description
          "Parameter communicated to the Control-Client as part of
           the Server Greeting message and used for deriving a key
           from a shared secret as per Section 3.1 of  RFC 4656:
           MUST be a power of 2 and at least 1024. It is configured
           by providing said power. For example, configuring 20 here
           means count 2^20 = 1048576. The default is 15,
           meaning 2^15 = 32768.";
        }
     description
       "Reusable data structure for count, which is used both in the
        Server and the Control-Client.";
    }

    grouping max-count-exponent {
      leaf max-count-exponent {
        type uint8 {
          range 10..31;
        }
        default 20;
        description
          "This parameter limits the maximum Count value, which MUST
           be a power of 2 and at least 1024 as per RFC 5357. It is
           configured by providing said power. For example,
           configuring 10 here means max count 2^10 = 1024.
           The default is 20, meaning 2^20 = 1048576.

           A TWAMP Server uses this configured value in the
           Server-Greeting message sent to the Control-Client.

           A TWAMP Control-Client uses this configured value to
           prevent denial-of-service (DOS) attacks by closing the
           control connection to the Server if it 'receives a
           Server-Greeting message with Count greater that its
           maximum configured value', as per Section 6 of RFC 5357.
```

```
            Further, note that according to Section 6 of RFC 5357:

            'If an attacking system sets the maximum value in
            Count (2**32), then the system under attack would stall
            for a significant period of time while it attempts to
            generate keys.

            TWAMP-compliant systems SHOULD have a configuration
            control to limit the maximum count value. The default
            max-count-exponent value SHOULD be 15 which corresponds
            to a maximum value of 2**15 or 32768.'

            RFC 5357 does not qualify 'significant period' in terms of
            time, but it is clear that this depends on the processing
            capacity available and operators need to pay attention to
            this security consideration.";
        }
      description
        "Reusable data structure for max-count which is used both at
         the Control-Client and the Server containers.";
    }


    /*
     * Configuration data nodes
     */

    container twamp {
      description
        "TWAMP logical entity configuration grouping of four models
        which correspond to the four TWAMP logical entities
        Control-Client, Server, Session-Sender, and Session-Reflector
        as illustrated in Fig. 1 of RFC XXXX.";

      container client {
        if-feature control-client;
        description
          "Configuration of the TWAMP Control-Client logical
           entity.";

        leaf admin-state {
          type boolean;
          default true;
          description
            "Indicates whether the device is allowed to operate as a
             TWAMP Control-Client.";
        }
```

```
        list mode-preference-chain {
          key priority;
          unique mode;
          leaf priority {
            type uint16;
            description
              "Indicates the Control-Client Mode preference priority
               expressed as a 16-bit unsigned integer. Values for the
               priority start with zero, the highest priority, and
               decreasing priority value is indicated by every increase
               in value by one.";
          }
          leaf mode {
            type twamp-modes;
            description
              "The supported TWAMP Mode matching the corresponding
               priority.";

          }
          description
            "Indicates the Control-Client preferred order of use of
             the supported TWAMP Modes.

             Depending on the Modes available in the TWAMP Server
             Greeting message (see Fig. 2 of RFC 7717), the
             Control-Client MUST choose the highest priority
             Mode from the configured mode-preference-chain list.";
        }

        uses key-management;

        list ctrl-connection {
          key name;
          description
            "List of TWAMP Control-Client control connections.
             Each item in the list describes a control connection
             that will be initiated by this Control-Client";

          leaf name {
            type string;
            description
              "A unique name used as a key to identify this
               individual TWAMP-Control connection on the
               Control-Client device.";
          }
          leaf client-ip {
            type inet:ip-address;
            description
```

```
            "The IP address of the local Control-Client device,
             to be placed in the source IP address field of the
             IP header in TWAMP-Control (TCP) packets belonging
             to this control connection. If not configured, the
             device SHALL choose its own source IP address.";
          }
          leaf server-ip {
            type inet:ip-address;
            mandatory true;
            description
              "The IP address of the remote Server device, which the
               TWAMP-Control connection will be initiated to.";
          }

          leaf server-tcp-port {
            type inet:port-number;
            default 862;
            description
              "This parameter defines the TCP port number that is
               to be used by this outgoing TWAMP-Control connection.
               Typically, this is the well-known TWAMP-Control
               port number (862) as per RFC 5357 However, there are
               known realizations of TWAMP in the field that were
               implemented before this well-known port number was
               allocated. These early implementations allowed the
               port number to be configured. This parameter is
               therefore provided for backward compatibility
               reasons.";
          }

          leaf control-packet-dscp {
            type inet:dscp;
            default 0;
            description
              "The DSCP value to be placed in the IP header of
               TWAMP-Control (TCP) packets generated by this
               Control-Client.";
          }

          leaf key-id {
            type string {
              length 1..80;
            }
            description
             "Indicates the KeyID value selected for this
              TWAMP-Control connection.";
          }
```

```
          uses max-count-exponent;

          leaf client-tcp-port {
            type inet:port-number;
            config false;
            description
              "Indicates the source TCP port number used in the
               TWAMP-Control packets belonging to this control
               connection.";
          }

          leaf server-start-time {
            type uint64;
            config false;
            description
              "Indicates the Start-Time advertised by the Server in
               the Server-Start message (RFC 4656, Section 3.1),
               representing the time when the current
               instantiation of the Server started operating.
               The timestamp format follows RFC 5905
               according to Section 4.1.2 of RFC 4656.";
            reference
              "RFC 4656: OWAMP, Section 3.1 and 4.1.2,
               RFC 5905: NTPv4 Specification.";
          }

          leaf repeat-count {
            type uint64;
            config false;
            description
              "Indicates how many times the test session has been
               repeated. When a test is running, this value will be
               greater than 0. If the repeat parameter is non-zero,
               this value is smaller than or equal to the repeat
               parameter.";
          }
          leaf state {
            type control-client-connection-state;
            config false;
            description
              "Indicates the current state of the TWAMP-Control
               connection state.";
          }

          leaf selected-mode {
            type twamp-modes;
            config false;
            description
```

```
            "The TWAMP Mode that the Control-Client has chosen for
             this control connection as set in the Mode field of
             the Set-Up-Response message";
          reference
            "RFC 4656, Section 3.1.";
        }

        leaf token {
            type binary {
            length 64;
          }
          config false;
          description
            "This parameter holds the 64 octets containing the
             concatenation of a 16-octet Challenge, a 16-octet AES
             Session-key used for encryption, and a 32-octet
             HMAC-SHA1 Session-key used for authentication; see
             also the last paragraph of Section 6 in RFC 4656.

             If the Mode defined in RFC 7717 is selected
             (selected-mode), Token is limited to 16 octets.";
          reference
            "RFC 4086: Randomness Requirements for Security

             RFC 7717: IKEv2-Derived Shared Secret Key for the
             One-Way Active Measurement Protocol (OWAMP) and
             Two-Way Active Measurement Protocol (TWAMP)";
        }

        leaf client-iv {
          type binary {
            length 16;
          }
          config false;
          description
            "Indicates the Control-Client Initialization Vector
             (Client-IV), that is generated randomly by the
             Control-Client. As per RFC 4656:

              Client-IV merely needs to be unique (i.e., it MUST
              never be repeated for different sessions using the
              same secret key; a simple way to achieve that without
              the use of cumbersome state is to generate the
              Client-IV values using a cryptographically secure
              pseudo-random number source.

              If the Mode defined in RFC 7717 is selected
              (selected-mode), Client-IV is limited to 12 octets.";
```

```
            reference
              "RFC 4656: A One-way Active Measurement Protocol
               (OWAMP).

               RFC 7717: IKEv2-Derived Shared Secret Key for the
               One-Way Active Measurement Protocol (OWAMP) and
               Two-Way Active Measurement Protocol (TWAMP)";
          }

          list test-session-request {
            key name;
            description
              "Information associated with the Control-Client
               for this test session";

            leaf name {
              type string;
              description
                "A unique name to be used for identification of
                 this TWAMP-Test session on the Control-Client.";
            }

            leaf sender-ip {
              type inet:ip-address;
              description
                "The IP address of the Session-Sender device,
                 which is to be placed in the source IP address
                 field of the IP header in TWAMP-Test (UDP) packets
                 belonging to this test session. This value will be
                 used to populate the sender address field of the
                 Request-TW-Session message.

                 If not configured, the device SHALL choose its own
                 source IP address.";
            }

            leaf sender-udp-port {
              type union {
                type dynamic-port-number;
                type enumeration {
                  enum autoallocate {
                    description
                      "Indicates that the Contol-Client will
                       auto-allocate the TWAMP-Test (UDP) port number
                       from the dynamic port range.";
                  }
                }
              }
```

```
        default autoallocate;
        description
          "The UDP port number that is to be used by
           the Session-Sender for this TWAMP-Test session.
           The number is restricted to the dynamic port range.

           By default the Control-Client SHALL auto-allocate a
           UDP port number for this TWAMP-Test session.

           The configured (or auto-allocated) value is
           advertised in the Sender Port field of the
           Request-TW-session message (see Section 3.5 of
           RFC 5357). Note that in the scenario where a device
           auto-allocates a UDP port number for a session, and
           the repeat parameter for that session indicates that
           it should be repeated, the device is free to
           auto-allocate a different UDP port number when it
           negotiates the next (repeated) iteration of this
           session.";
      }

      leaf reflector-ip {
        type inet:ip-address;
        mandatory true;
        description
          "The IP address belonging to the remote
           Session-Reflector device to which the TWAMP-Test
           session will be initiated. This value will be
           used to populate the receiver address field of
           the Request-TW-Session message.";
      }

      leaf reflector-udp-port {
        type inet:port-number {
          range "862 | 49152..65535";
        }
        description
          "This parameter defines the UDP port number that
           will be used by the Session-Reflector for
           this TWAMP-Test session. The default number is
           within the dynamic port range and is to be placed
           in the Receiver Port field of the Request-TW-Session
           message. The well-known port (862) MAY be
           used.";
        reference
          "draft-ietf-ippm-port-twamp-test: OWAMP and TWAMP
           Well-Known Port Assignments.";
      }
```

```
            leaf timeout {
              type uint64;
              units seconds;
              default 2;
              description
                "The length of time (in seconds) that the
                 Session-Reflector should continue to respond to
                 packets belonging to this TWAMP-Test session after
                 a Stop-Sessions TWAMP-Control message has been
                 received.

                 This value will be placed in the Timeout field of
                 the Request-TW-Session message.";
              reference
                "RFC 5357: TWAMP, Section 3.5.";
            }

            leaf padding-length {
              type uint32 {
                range 64..4096;
              }
              description
                  "The number of padding bytes to be added to the
                   TWAMP-Test (UDP) packets generated by the
                   Session-Sender.

                   This value will be placed in the Padding Length
                   field of the Request-TW-Session message.";
              reference
                "RFC 4656, Section 3.5.";
            }

            leaf test-packet-dscp {
              type inet:dscp;
              default 0;
              description
                "The DSCP value to be placed in the IP header
                 of TWAMP-Test packets generated by the
                 Session-Sender, and in the UDP header of the
                 TWAMP-Test response packets generated by the
                 Session-Reflector for this test session.

                 This value will be placed in the Type-P Descriptor
                 field of the Request-TW-Session message";
              reference
                "RFC 5357.";
            }
```

```
            leaf start-time {
              type uint64;
              default 0;
              description
                "Time when the session is to be started
                 (but not before the TWAMP Start-Sessions command
                 is issued; see Section 3.4 of RFC 5357).

                 The start-time value is placed in the Start Time
                 field of the Request-TW-Session message.

                 The timestamp format follows RFC 5905 as per
                 Section 3.5 of RFC 4656.

                 The default value of 0 indicates that the session
                 will be started as soon as the Start-Sessions
                 message is received.";
            }

            leaf repeat {
              type uint32 {
                range 0..4294967295;
              }
              default 0;
              description
                "This value determines if the TWAMP-Test session must
                 be repeated. When a test session has completed, the
                 repeat parameter is checked.

                 The default value of 0 indicates that the session
                 MUST NOT be repeated.

                 If the repeat value is 1 through 4,294,967,294
                 then the test session SHALL be repeated using the
                 information in repeat-interval parameter, and the
                 parent TWAMP-Control connection for this test
                 session is restarted to negotiate a new instance
                 of this TWAMP-Test session.

                 A value of 4,294,967,295 indicates that the test
                 session SHALL be repeated *forever* using the
                 information in repeat-interval parameter, and SHALL
                 NOT decrement the value.";
            }

            leaf repeat-interval  {
              when "../repeat!='0'" {
                description
```

```
                "This parameter determines the timing of repeated
                 TWAMP-Test sessions when repeat is more than 0.

                 When the value of repeat-interval is 0, the
                 negotiation of a new test session SHALL begin
                 immediately after the previous test session
                 completes. Otherwise, the Control-Client will
                 wait for the number of seconds specified in the
                 repeat-interval parameter before negotiating the
                 new instance of this TWAMP-Test session.";
            }
            type uint32;
            units seconds;
            default 0;
            description
              "Repeat interval (in seconds).";
          }

          list pm-reg-list {
            key pm-index;
            leaf pm-index {
              type uint16;
              description
                "Numerical index value of a Registered Metric
                 in the Performance Metric Registry
                 (see ietf-ippm-metric-registry). Output statistics
                 are specified in the corresponding Registry
                 entry.";
            }
            description
              "A list of one or more Performance Metric Registry
               Index values, which communicate packet stream
               characteristics along with one or more metrics
               to be measured.

               All members of the pm-reg-list MUST have the same
               stream characteristics, such that they combine
               to specify all metrics that shall be measured on
               a single stream.";
            reference
              "ietf-ippm-metric-registry: Registry for
               Performance Metrics";
          }

          leaf state {
            type test-session-state;
            config false;
            description
```

```
                "Indicates the TWAMP-Test session state, accepted or
                 indication of an error.";
              reference
                "Section 3.5 of RFC 5357.";
            }
            leaf sid {
              type string;
              config false;
              description
                "The SID allocated by the Server for this TWAMP-Test
                 session, and communicated back to the Control-Client
                 in the SID field of the Accept-Session message";
              reference
                "Section 4.3 of RFC 6038.";
            }
          }
        }
      }

      container server {
        if-feature server;
        description
          "Configuration of the TWAMP Server logical entity.";

        leaf admin-state {
          type boolean;
          default true;
          description
            "Indicates whether the device is allowed to operate
             as a TWAMP Server.";
        }

        leaf server-tcp-port {
          type inet:port-number;
          default 862;
          description
            "This parameter defines the well known TCP port number
             that is used by TWAMP-Control. The Server will listen
             on this port number for incoming TWAMP-Control
             connections. Although this is defined as a fixed value
             (862) in RFC 5357, there are several realizations of
             TWAMP in the field that were implemented before this
             well-known port number was allocated. These early
             implementations allowed the port number to be
             configured. This parameter is therefore provided for
             backward compatibility reasons.";
        }
```

```
        leaf servwait {
          type uint32 {
            range 1..604800;
          }
          units seconds;
          default 900;
          description
            "TWAMP-Control (TCP) session timeout, in seconds.
             According to Section 3.1 of RFC 5357,

             Server MAY discontinue any established control
             connection when no packet associated with that
             connection has been received within SERVWAIT seconds.";
        }

        leaf control-packet-dscp {
          type inet:dscp;
          description
            "The DSCP value to be placed in the IP header of
             TWAMP-Control (TCP) packets generated by the Server.

             Section 3.1 of  RFC 5357 specifies that the server
             SHOULD use the DSCP value from the Control-Clients
             TCP SYN. However, for practical purposes TWAMP will
             typically be implemented using a general purpose TCP
             stack provided by the underlying operating system,
             and such a stack may not provide this information to the
             user. Consequently, it is not always possible to
             implement the behavior described in RFC 5357 in an
             OS-portable version of TWAMP.

             The default behavior if this item is not set is to use
             the DSCP value from the Control-Clients TCP SYN.";
           reference
             "Section 3.1 of RFC 5357.";
        }

        uses count;

        uses max-count-exponent;

        leaf modes {
          type twamp-modes;
          description
            "The bit mask of TWAMP Modes this Server instance
             is willing to support; see IANA TWAMP Modes Registry.";
        }
```

```
        uses key-management;

        list ctrl-connection {
          key "client-ip client-tcp-port server-ip server-tcp-port";
          config false;
          description
            "List of all incoming TWAMP-Control (TCP) connections.";

          leaf client-ip {
            type inet:ip-address;
            description
              "The IP address on the remote Control-Client device,
               which is the source IP address used in the
               TWAMP-Control (TCP) packets belonging to this control
               connection.";
          }

          leaf client-tcp-port {
            type inet:port-number;
            description
              "The source TCP port number used in the TWAMP-Control
               (TCP) packets belonging to this control connection.";
          }

          leaf server-ip {
            type inet:ip-address;
            description
              "The IP address of the local Server device, which is
               the destination IP address used in the
               TWAMP-Control (TCP) packets belonging to this control
               connection.";
          }

          leaf server-tcp-port {
            type inet:port-number;
            description
              "The destination TCP port number used in the
               TWAMP-Control (TCP) packets belonging to this
               control connection. This will usually be the
               same value as the server-tcp-port configured
               under twamp/server. However, in the event that
               the user re-configured server/server-tcp-port
               after this control connection was initiated, this
               value will indicate the server-tcp-port that is
               actually in use for this control connection.";
          }

          leaf state {
```

```
            type server-ctrl-connection-state;
            description
              "Indicates the Server TWAMP-Control connection state.";
          }

          leaf control-packet-dscp {
            type inet:dscp;
            description
              "The DSCP value used in the IP header of the
               TWAMP-Control (TCP) packets sent by the Server
               for this control connection. This will usually
               be the same value as is configured in the
               control-packet-dscp parameter under the twamp/server
               container.  However, in the event that the user
               re-configures server/dscp after this control
               connection is already in progress, this read-only
               value will show the actual dscp value in use by this
               TWAMP-Control connection.";
          }

          leaf selected-mode {
            type twamp-modes;
            description
              "The Mode that was chosen for this TWAMP-Control
               connection as set in the Mode field of the
               Set-Up-Response message.";
          }

          leaf key-id {
            type string {
              length 1..80;
            }
            description
              "The KeyID value that is in use by this TWAMP-Control
               connection as selected by Control-Client.";
          }

          uses count {
            description
              "The count value that is in use by this TWAMP-Control
               connection. This will usually be the same value
               as is configured under twamp/server. However, in the
               event that the user re-configured server/count
               after this control connection is already in progress,
               this read-only value will show the actual count that
               is in use for this TWAMP-Control connection.";
          }
```

```
          uses max-count-exponent {
            description
              "This read-only value indicates the actual max-count in
               use for this control connection. Usually this would be
               the same value as configured under twamp/server.";
          }

          leaf salt {
            type binary {
              length 16;
            }
            description
              "A parameter used in deriving a key from a
               shared secret as described in Section 3.1 of RFC 4656.
               It is communicated to the Control-Client as part of
               the Server Greeting message.";
          }

          leaf server-iv {
            type binary {
              length 16;
            }
            description
              "The Server Initialization Vector
               (IV) generated randomly by the Server.";
          }

          leaf challenge {
            type binary {
              length 16;
            }
            description
              "A random sequence of octets generated by the Server.
               As described in client/token, Challenge is used
               by the Control-Client to prove possession of a
               shared secret.";
          }
        }
      }

      container session-sender {
        if-feature session-sender;
        description
          "Configuration of the TWAMP Session-Sender logical entity";
        leaf admin-state {
          type boolean;
          default true;
          description
```

```
            "Indicates whether the device is allowed to operate
             as a TWAMP Session-Sender.";
          }

          list test-session{
            key name;
            description
              "List of TWAMP Session-Sender test sessions.";

            leaf name {
              type string;
              description
                "A unique name for this TWAMP-Test session to be used
                 for identifying this test session by the
                 Session-Sender logical entity.";
            }

            leaf ctrl-connection-name {
              type string;
              config false;
              description
                "The name of the parent TWAMP-Control connection that
                 is responsible for negotiating this TWAMP-Test
                 session.";
            }

            leaf fill-mode {
              type padding-fill-mode;
              default zero;
              description
                "Indicates whether the padding added to the
                 TWAMP-Test (UDP) packets will contain pseudo-random
                 numbers, or whether it should consist of all zeroes,
                 as per Section 4.2.1 of RFC 5357.";
            }

            leaf number-of-packets {
              type uint32;
              mandatory true;
              description
                "The overall number of TWAMP-Test (UDP) packets to be
                 transmitted by the Session-Sender for this test
                 session.";
            }

            choice packet-distribution {
              description
                "Indicates the distribution to be used for transmitting
```

```
               the TWAMP-Test (UDP) packets.";
             case periodic {
               leaf periodic-interval {
                 type decimal64 {
                   fraction-digits 5;
                 }
                 units seconds;
                 mandatory true;
                 description
                   "Indicates the time to wait (in seconds) between
                    the first bits of TWAMP-Test (UDP) packet
                    transmissions for this test session.";
                 reference
                   "RFC 3432: Network performance measurement
                              with periodic streams";
               }
             }
             case poisson {
               leaf lambda {
                 type decimal64 {
                   fraction-digits 5;
                 }
                 units seconds;
                 mandatory true;
                 description
                   "Indicates the average time interval (in seconds)
                    between packets in the Poisson distribution.
                    The packet is calculated using the reciprocal of
                    lambda and the TWAMP-Test packet size (which
                    depends on the selected Mode and the packet
                    padding).";
                reference
                  "RFC 2330: Framework for IP Performance Metrics";
               }
               leaf max-interval {
                 type decimal64 {
                   fraction-digits 5;
                 }
                 units seconds;
                 description
                   "Indicates the maximum time (in seconds)
                    between packet transmissions.";
                 reference
                   "RFC 7312: Advanced Stream and Sampling Framework
                              for IP Performance Metrics (IPPM)";
               }
             }
           }
```

```
          leaf state {
            type sender-session-state;
            config false;
            description
              "Indicates the Session-Sender test session state.";
          }

          uses maintenance-statistics;
        }
      }

      container session-reflector {
        if-feature session-reflector;
        description
          "Configuration of the TWAMP Session-Reflector logical
           entity";

        leaf admin-state {
          type boolean;
          default true;
          description
            "Indicates whether the device is allowed to operate
             as a TWAMP Session-Reflector.";
        }

        leaf refwait {
          type uint32 {
            range 1..604800;
          }
          units seconds;
          default 900;
          description
            "The Session-Reflector MAY discontinue any session that
             has been started when no packet associated with that
             session has been received for REFWAIT seconds. As per
             Section 3.1 of RFC 5357, this timeout allows a
             Session-Reflector to free up resources in case of
             failure.";
        }

        list test-session {
          key
            "sender-ip sender-udp-port
             reflector-ip reflector-udp-port";
          config false;
          description
            "TWAMP Session-Reflectortest sessions.";
```

```
          leaf sid {
            type string;
            description
              "An auto-allocated identifier for this TWAMP-Test
               session that is unique within the context of this
               Server/Session-Reflector device only. This value
               is communicated to the Control-Client that
               requested the test session in the SID field of the
               Accept-Session message.";
          }

          leaf sender-ip {
            type inet:ip-address;
            description
              "The IP address on the remote device, which is the
               source IP address used in the TWAMP-Test (UDP) packets
               belonging to this test session.";
          }

          leaf sender-udp-port {
            type dynamic-port-number;
            description
              "The source UDP port used in the TWAMP-Test packets
               belonging to this test session.";
          }

          leaf reflector-ip {
            type inet:ip-address;
            description
              "The IP address of the local Session-Reflector
               device, which is the destination IP address used
               in the TWAMP-Test (UDP) packets belonging to this test
               session.";
          }

          leaf reflector-udp-port {
            type inet:port-number {
              range "862 | 49152..65535";
              }
            description
              "The destination UDP port number used in the
               TWAMP-Test (UDP) test packets belonging to this
               test session.";
          }

          leaf parent-connection-client-ip {
            type inet:ip-address;
            description
```

```
              "The IP address on the Control-Client device, which
               is the source IP address used in the TWAMP-Control
               (TCP) packets belonging to the parent control
               connection that negotiated this test session.";
            }

            leaf parent-connection-client-tcp-port {
              type inet:port-number;
              description
                "The source TCP port number used in the TWAMP-Control
                 (TCP) packets belonging to the parent control
                 connection that negotiated this test session.";
            }

            leaf parent-connection-server-ip {
              type inet:ip-address;
              description
                "The IP address of the Server device, which is the
                 destination IP address used in the TWAMP-Control
                 (TCP) packets belonging to the parent control
                 connection that negotiated this test session.";
            }

            leaf parent-connection-server-tcp-port {
              type inet:port-number;
              description
                "The destination TCP port number used in the
                 TWAMP-Control (TCP) packets belonging to the parent
                 control connection that negotiated this test
                 session.";
            }

            leaf test-packet-dscp {
              type inet:dscp;
              description
                "The DSCP value present in the IP header of
                 TWAMP-Test (UDP) packets belonging to this session.";
            }

            uses maintenance-statistics;
          }
        }
      }
    }

    <CODE ENDS>
```

6. Data Model Examples

This section presents a simple but complete example of configuring all four entities in Figure 1, based on the YANG module specified in Section 5. The example is illustrative in nature, but aims to be self-contained, i.e. were it to be executed in a real TWAMP implementation it would lead to a correctly configured test session. For completeness, examples are provided for both IPv4 and IPv6.

A more elaborated example, which also includes authentication parameters, is provided in Appendix A.

6.1. Control-Client

Figure 8 shows a configuration example for a Control-Client with client/admin-state enabled. In a real implementation following Figure 2 this would permit the initiation of TWAMP-Control connections and TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <client>
      <admin-state>true</admin-state>
    </client>
  </twamp>
</config>
```

Figure 8: XML instance enabling Control-Client operation.

The following example shows a Control-Client with two instances of client/ctrl-connection, one called "RouterA" and another called "RouterB". Each TWAMP-Control connection is to a different Server. The control connection named "RouterA" has two test session requests. The TWAMP-Control connection named "RouterB" has no TWAMP-Test session requests.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <client>
      <admin-state>true</admin-state>
      <ctrl-connection>
        <name>RouterA</name>
        <client-ip>203.0.113.1</client-ip>
        <server-ip>203.0.113.2</server-ip>
        <test-session-request>
```

```
            <name>Test1</name>
            <sender-ip>203.0.113.3</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>203.0.113.4</reflector-ip>
            <reflector-udp-port>50001</reflector-udp-port>
            <start-time>0</start-time>
          </test-session-request>
          <test-session-request>
            <name>Test2</name>
            <sender-ip>203.0.113.1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>203.0.113.2</reflector-ip>
            <reflector-udp-port>50001</reflector-udp-port>
            <start-time>0</start-time>
          </test-session-request>
        </ctrl-connection>
        <ctrl-connection>
          <name>RouterB</name>
          <client-ip>203.0.113.1</client-ip>
          <server-ip>203.0.113.3</server-ip>
        </ctrl-connection>
      </client>
    </twamp>
  </config>

  <?xml version="1.0" encoding="utf-8"?>
  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <client>
        <admin-state>true</admin-state>
        <ctrl-connection>
          <name>RouterA</name>
          <client-ip>2001:DB8:203:0:113::1</client-ip>
          <server-ip>2001:DB8:203:0:113::2</server-ip>
          <test-session-request>
            <name>Test1</name>
            <sender-ip>2001:DB8:203:1:113::3</sender-ip>
            <sender-udp-port>54000</sender-udp-port>
            <reflector-ip>2001:DB8:203:1:113::4</reflector-ip>
            <reflector-udp-port>55000</reflector-udp-port>
            <start-time>0</start-time>
          </test-session-request>
          <test-session-request>
            <name>Test2</name>
            <sender-ip>2001:DB8:203:0:113::1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
            <reflector-udp-port>55001</reflector-udp-port>
```

```
            <start-time>0</start-time>
          </test-session-request>
        </ctrl-connection>
        <ctrl-connection>
          <name>RouterB</name>
          <client-ip>2001:DB8:203:0:113::1</client-ip>
          <server-ip>2001:DB8:203:0:113::3</server-ip>
        </ctrl-connection>
      </client>
    </twamp>
  </config>
```

6.2. Server

Figure 9 shows a configuration example for a Server with server/admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Control connections and TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <server>
      <admin-state>true</admin-state>
    </server>
  </twamp>
</config>
```

Figure 9: XML instance enabling Server operation.

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

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <server>
      <admin-state>true</admin-state>
      <ctrl-connection>
        <client-ip>203.0.113.1</client-ip>
        <client-tcp-port>16341</client-tcp-port>
        <server-ip>203.0.113.2</server-ip>
        <server-tcp-port>862</server-tcp-port>
        <state>active</state>
      </ctrl-connection>
    </server>
  </twamp>
</data>

<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <server>
      <admin-state>true</admin-state>
      <ctrl-connection>
        <client-ip>2001:DB8:203:0:113::1</client-ip>
        <client-tcp-port>16341</client-tcp-port>
        <server-ip>2001:DB8:203:0:113::2</server-ip>
        <server-tcp-port>862</server-tcp-port>
        <state>active</state>
      </ctrl-connection>
    </server>
  </twamp>
</data>
```

6.3. Session-Sender

Figure 10 shows a configuration example for a Session-Sender with session-sender/admin-state enabled, which permits a device following Figure 2 to initiate TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-sender>
      <admin-state>true</admin-state>
    </session-sender>
  </twamp>
</config>
```

Figure 10: XML instance enabling Session-Sender operation.

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

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-sender>
      <admin-state>true</admin-state>
      <test-session>
        <name>Test1</name>
        <ctrl-connection-name>RouterA</ctrl-connection-name>
        <number-of-packets>900</number-of-packets>
        <periodic-interval>1</periodic-interval>
      </test-session>
      <test-session>
        <name>Test2</name>
        <ctrl-connection-name>RouterA</ctrl-connection-name>
        <number-of-packets>900</number-of-packets>
        <lambda>1</lambda>
        <max-interval>2</max-interval>
      </test-session>
    </session-sender>
  </twamp>
</data>
```

6.4. Session-Reflector

This configuration example shows a Session-Reflector with session-reflector/admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
    </session-reflector>
  </twamp>
</config>
```

Figure 11: XML instance enabling Session-Reflector operation.

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

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
      <test-session>
        <sender-ip>203.0.113.3</sender-ip>
        <sender-udp-port>54000</sender-udp-port>
        <reflector-ip>203.0.113.4</reflector-ip>
        <reflector-udp-port>50001</reflector-udp-port>
        <sid>1232</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <sent-packets>2</sent-packets>
        <rcv-packets>2</rcv-packets>
        <last-sent-seq>1</last-sent-seq>
        <last-rcv-seq>1</last-rcv-seq>
      </test-session>
      <test-session>
        <sender-ip>203.0.113.1</sender-ip>
        <sender-udp-port>54001</sender-udp-port>
        <reflector-ip>192.0.2.2</reflector-ip>
        <reflector-udp-port>50001</reflector-udp-port>
        <sid>178943</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
```

```
   client-ip>
           <parent-connection-client-tcp-port>16341</parent-connection-\
   client-tcp-port>
           <parent-connection-server-ip>203.0.113.2</parent-connection-\
   server-ip>
           <parent-connection-server-tcp-port>862</parent-connection-se\
   rver-tcp-port>
           <sent-packets>21</sent-packets>
           <rcv-packets>21</rcv-packets>
           <last-sent-seq>20</last-sent-seq>
           <last-rcv-seq>20</last-rcv-seq>
         </test-session>
       </session-reflector>
     </twamp>
   </data>

   [note: '\' line wrapping is for formatting only]

   <?xml version="1.0" encoding="utf-8"?>
   <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
       <session-reflector>
         <admin-state>true</admin-state>
         <test-session>
           <sender-ip>203.0.113.3</sender-ip>
           <sender-udp-port>54000</sender-udp-port>
           <reflector-ip>203.0.113.4</reflector-ip>
           <reflector-udp-port>54001</reflector-udp-port>
           <sid>1232</sid>
           <parent-connection-client-ip>203.0.113.1</parent-connection-\
   client-ip>
           <parent-connection-client-tcp-port>16341</parent-connection-\
   client-tcp-port>
           <parent-connection-server-ip>203.0.113.2</parent-connection-\
   server-ip>
           <parent-connection-server-tcp-port>862</parent-connection-se\
   rver-tcp-port>
           <sent-packets>2</sent-packets>
           <rcv-packets>2</rcv-packets>
           <last-sent-seq>1</last-sent-seq>
           <last-rcv-seq>1</last-rcv-seq>
         </test-session>
         <test-session>
           <sender-ip>203.0.113.1</sender-ip>
           <sender-udp-port>54001</sender-udp-port>
           <reflector-ip>192.0.2.2</reflector-ip>
           <reflector-udp-port>55001</reflector-udp-port>
           <sid>178943</sid>
```

```
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <sent-packets>21</sent-packets>
        <rcv-packets>21</rcv-packets>
        <last-sent-seq>20</last-sent-seq>
        <last-rcv-seq>20</last-rcv-seq>
      </test-session>
    </session-reflector>
  </twamp>
</data>
```

7. Security Considerations

Virtually all existing measurement systems using TWAMP [RFC5357] are administered by the same network operator. Attacks on the measurement infrastructure could be launched by third-parties to commandeer the packet generation capability, corrupt the measurements, or other examples of nefarious acts.

The YANG module specified in Section 5 of this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF [RFC6241] layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF Access Control Module (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of nodes defined in this YANG module which are writeable. These data nodes may be considered sensitive and vulnerable to attacks in some network environments. Ability to write into these nodes without proper protection can have a negative effect on the devices that support this feature.

If written, the 'admin-state' node can cause unintended test sessions to be created. If the node 'number-of-packets' that dictates how many packets are sent in any particular test session is written with

a large value, it can cause a test session to run longer than expected. Nodes that are particularly vulnerable include several timeout values put in the protocol to protect against sessions that are not active but are consuming resources. These are the REFWAIT timeout parameter which determine whether to discontinue the session if no packets are received, and nodes 'count' and 'max-count-exponent' which can cause a long time to be spent on PBKDF2 iterations. In addition, 'dscp' node marked with different DSCP markings, can cause the test traffic on the network to be skewed, and the result manipulated. Finally, nodes within 'mode-preference-chain' which specify the 'mode' and 'priority' values and indicate the preferred order of use by an operator, can be manipulated to send unauthenticated or non-encrypted traffic, enabling a MITM attack. Limiting access to these nodes will limit the ability to launch an attack in network environments.

The 'token' node defined in the model, containing a concatenation of a Challenge, AES Session-key used for encryption, and HMAC-SHA1 Session-key used for authentication, is sensitive from a privacy perspective, and can be used to disrupt a test session. The ability to read the field should be limited to the administrator of the test network.

8. IANA Considerations

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

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

Registrant Contact: The IESG.

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

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

name: ietf-twamp

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

prefix: twamp

reference: RFC XXXX

9. Acknowledgements

We thank Fred Baker, Kevin D'Souza, Gregory Mirsky, Brian Trammell, Robert Sherman, and Marius Georgescu for their thorough and constructive reviews, comments and text suggestions.

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

Jan Lindblad and Ladislav Lhokta did thorough reviews of the YANG module and the examples in Appendix A.

Kostas Pentikousis was partially supported by FP7 UNIFY (http://fp7-unify.eu), a research project partially funded by the European Community under the Seventh Framework Program (grant agreement no. 619609). The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

10. Contributors

Lianshu Zheng.

11. References

11.1. Normative References

[I-D.ietf-ippm-metric-registry]
Bagnulo, M., Claise, B., Eardley, P., Morton, A., and A. Akhter, "Registry for Performance Metrics", draft-ietf-ippm-metric-registry-14 (work in progress), March 2018.

[I-D.ietf-ippm-port-twamp-test]
Morton, A. and G. Mirsky, "OWAMP and TWAMP Well-Known Port Assignments", draft-ietf-ippm-port-twamp-test-01 (work in progress), March 2018.

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

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

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

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

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

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

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

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

[RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features", RFC 6038, DOI 10.17487/RFC6038, October 2010, <https://www.rfc-editor.org/info/rfc6038>.

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

[RFC7717] Pentikousis, K., Ed., Zhang, E., and Y. Cui, "IKEv2-Derived Shared Secret Key for the One-Way Active Measurement Protocol (OWAMP) and Two-Way Active Measurement Protocol (TWAMP)", RFC 7717, DOI 10.17487/RFC7717, December 2015, <https://www.rfc-editor.org/info/rfc7717>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[UML] ISO/IEC, "Information technology - Open Distributed Processing - Unified Modeling Language", April 2005.

11.2. Informative References

[NSC] John, W., Pentikousis, K., et al., "Research directions in network service chaining", Proc. SDN for Future Networks and Services (SDN4FNS), Trento, Italy IEEE, November 2013.

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5618] Morton, A. and K. Hedayat, "Mixed Security Mode for the Two-Way Active Measurement Protocol (TWAMP)", RFC 5618, DOI 10.17487/RFC5618, August 2009, <https://www.rfc-editor.org/info/rfc5618>.

[RFC5938] Morton, A. and M. Chiba, "Individual Session Control Feature for the Two-Way Active Measurement Protocol (TWAMP)", RFC 5938, DOI 10.17487/RFC5938, August 2010, <https://www.rfc-editor.org/info/rfc5938>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <https://www.rfc-editor.org/info/rfc7426>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. Detailed Data Model Examples

This appendix extends the example presented in Section 6 by configuring more fields such as authentication parameters, DSCP values and so on.

A.1. Control-Client

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <client>
      <admin-state>true</admin-state>
      <mode-preference-chain>
        <priority>0</priority>
        <mode>authenticated</mode>
      </mode-preference-chain>
      <mode-preference-chain>
        <priority>1</priority>
```

```
          <mode>unauthenticated</mode>
        </mode-preference-chain>
        <key-chain>
          <key-id>KeyClient1ToRouterA</key-id>
          <secret-key>c2VjcmV0MQ==</secret-key>
        </key-chain>
        <key-chain>
          <key-id>KeyForRouterB</key-id>
          <secret-key>c2VjcmV0Mg0K</secret-key>
        </key-chain>
        <ctrl-connection>
          <name>RouterA</name>
          <client-ip>203.0.113.1</client-ip>
          <server-ip>203.0.113.2</server-ip>
          <control-packet-dscp>32</control-packet-dscp>
          <key-id>KeyClient1ToRouterA</key-id>
          <test-session-request>
            <name>Test1</name>
            <sender-ip>203.0.113.3</sender-ip>
            <sender-udp-port>54000</sender-udp-port>
            <reflector-ip>203.0.113.4</reflector-ip>
            <reflector-udp-port>55000</reflector-udp-port>
            <padding-length>64</padding-length>
            <start-time>0</start-time>
          </test-session-request>
          <test-session-request>
            <name>Test2</name>
            <sender-ip>203.0.113.1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>203.0.113.2</reflector-ip>
            <reflector-udp-port>55001</reflector-udp-port>
            <padding-length>128</padding-length>
            <start-time>0</start-time>
          </test-session-request>
        </ctrl-connection>
      </client>
    </twamp>
  </data>

  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <client>
        <admin-state>true</admin-state>
        <mode-preference-chain>
          <priority>0</priority>
          <mode>authenticated</mode>
        </mode-preference-chain>
```

```
        <mode-preference-chain>
          <priority>1</priority>
          <mode>unauthenticated</mode>
        </mode-preference-chain>
        <key-chain>
          <key-id>KeyClient1ToRouterA</key-id>
          <secret-key>c2VjcmV0MQ==</secret-key>
        </key-chain>
        <key-chain>
          <key-id>KeyForRouterB</key-id>
          <secret-key>c2VjcmV0Mg0K</secret-key>
        </key-chain>
        <ctrl-connection>
          <name>RouterA</name>
          <client-ip>2001:DB8:203:0:113::1</client-ip>
          <server-ip>2001:DB8:203:0:113::2</server-ip>
          <control-packet-dscp>32</control-packet-dscp>
          <key-id>KeyClient1ToRouterA</key-id>
          <test-session-request>
            <name>Test1</name>
            <sender-ip>2001:DB8:10:1:1::1</sender-ip>
            <sender-udp-port>54000</sender-udp-port>
            <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
            <reflector-udp-port>55000</reflector-udp-port>
            <padding-length>64</padding-length>
            <start-time>0</start-time>
          </test-session-request>
          <test-session-request>
            <name>Test2</name>
            <sender-ip>2001:DB8:203:0:113::1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
            <reflector-udp-port>55001</reflector-udp-port>
            <padding-length>128</padding-length>
            <start-time>0</start-time>
          </test-session-request>
        </ctrl-connection>
      </client>
    </twamp>
  </data>
```

A.2. Server

```
  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <server>
```

```
        <admin-state>true</admin-state>
        <servwait>1800</servwait>
        <control-packet-dscp>32</control-packet-dscp>
        <modes>authenticated unauthenticated</modes>
        <count>15</count>
        <key-chain>
          <key-id>KeyClient1ToRouterA</key-id>
          <secret-key>c2VjcmV0MQ==</secret-key>
        </key-chain>
        <key-chain>
          <key-id>KeyClient10ToRouterA</key-id>
          <secret-key>c2VjcmV0MTANCg==</secret-key>
        </key-chain>
        <ctrl-connection>
          <client-ip>203.0.113.1</client-ip>
          <client-tcp-port>16341</client-tcp-port>
          <server-ip>203.0.113.2</server-ip>
          <server-tcp-port>862</server-tcp-port>
          <control-packet-dscp>32</control-packet-dscp>
          <selected-mode>unauthenticated</selected-mode>
          <key-id>KeyClient1ToRouterA</key-id>
          <count>15</count>
        </ctrl-connection>
      </server>
    </twamp>
  </data>

  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <server>
        <admin-state>true</admin-state>
        <servwait>1800</servwait>
        <control-packet-dscp>32</control-packet-dscp>
        <modes>authenticated unauthenticated</modes>
        <count>15</count>
        <key-chain>
          <key-id>KeyClient1ToRouterA</key-id>
          <secret-key>c2VjcmV0MQ==</secret-key>
        </key-chain>
        <key-chain>
          <key-id>KeyClient10ToRouterA</key-id>
          <secret-key>c2VjcmV0MTANCg==</secret-key>
        </key-chain>
        <ctrl-connection>
          <client-ip>2001:DB8:203:0:113::1</client-ip>
          <client-tcp-port>16341</client-tcp-port>
          <server-ip>2001:DB8:203:0:113::2</server-ip>
```

```
          <server-tcp-port>862</server-tcp-port>
          <control-packet-dscp>32</control-packet-dscp>
          <selected-mode>unauthenticated</selected-mode>
          <key-id>KeyClient1ToRouterA</key-id>
          <count>15</count>
        </ctrl-connection>
      </server>
    </twamp>
  </data>
```

A.3. Session-Sender

```
  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <session-sender>
        <admin-state>true</admin-state>
        <test-session>
          <name>Test1</name>
          <ctrl-connection-name>RouterA</ctrl-connection-name>
          <fill-mode>zero</fill-mode>
          <number-of-packets>900</number-of-packets>
          <periodic-interval>1</periodic-interval>
          <sent-packets>2</sent-packets>
          <rcv-packets>2</rcv-packets>
          <last-sent-seq>1</last-sent-seq>
          <last-rcv-seq>1</last-rcv-seq>
        </test-session>
        <test-session>
          <name>Test2</name>
          <ctrl-connection-name>RouterA</ctrl-connection-name>
          <fill-mode>random</fill-mode>
          <number-of-packets>900</number-of-packets>
          <lambda>1</lambda>
          <max-interval>2</max-interval>
          <sent-packets>21</sent-packets>
          <rcv-packets>21</rcv-packets>
          <last-sent-seq>20</last-sent-seq>
          <last-rcv-seq>20</last-rcv-seq>
        </test-session>
      </session-sender>
    </twamp>
  </data>
```

A.4. Session-Reflector

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
      <test-session>
        <sender-ip>203.0.113.3</sender-ip>
        <sender-udp-port>54000</sender-udp-port>
        <reflector-ip>203.0.113.4</reflector-ip>
        <reflector-udp-port>55000</reflector-udp-port>
        <sid>1232</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <test-packet-dscp>32</test-packet-dscp>
        <sent-packets>2</sent-packets>
        <rcv-packets>2</rcv-packets>
        <last-sent-seq>1</last-sent-seq>
        <last-rcv-seq>1</last-rcv-seq>
      </test-session>
      <test-session>
        <sender-ip>203.0.113.1</sender-ip>
        <sender-udp-port>54001</sender-udp-port>
        <reflector-ip>192.0.2.2</reflector-ip>
        <reflector-udp-port>55001</reflector-udp-port>
        <sid>178943</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <test-packet-dscp>32</test-packet-dscp>
        <sent-packets>21</sent-packets>
        <rcv-packets>21</rcv-packets>
        <last-sent-seq>20</last-sent-seq>
        <last-rcv-seq>20</last-rcv-seq>
```

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

  [note: '\' line wrapping is for formatting only]

  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <session-reflector>
        <admin-state>true</admin-state>
        <test-session>
          <sender-ip>2001:DB8:10:1:1::1</sender-ip>
          <sender-udp-port>54000</sender-udp-port>
          <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
          <reflector-udp-port>55000</reflector-udp-port>
          <sid>1232</sid>
          <parent-connection-client-ip>2001:DB8:203:0:113::1</parent-c\
  onnection-client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
  client-tcp-port>
          <parent-connection-server-ip>2001:DB8:203:0:113::2</parent-c\
  onnection-server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
  rver-tcp-port>
          <test-packet-dscp>32</test-packet-dscp>
          <sent-packets>2</sent-packets>
          <rcv-packets>2</rcv-packets>
          <last-sent-seq>1</last-sent-seq>
          <last-rcv-seq>1</last-rcv-seq>
        </test-session>
        <test-session>
          <sender-ip>2001:DB8:203:0:113::1</sender-ip>
          <sender-udp-port>54001</sender-udp-port>
          <reflector-ip>2001:DB8:192:68::2</reflector-ip>
          <reflector-udp-port>55001</reflector-udp-port>
          <sid>178943</sid>
          <parent-connection-client-ip>2001:DB8:203:0:113::1</parent-c\
  onnection-client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
  client-tcp-port>
          <parent-connection-server-ip>2001:DB8:203:0:113::2</parent-c\
  onnection-server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
  rver-tcp-port>
          <test-packet-dscp>32</test-packet-dscp>
          <sent-packets>21</sent-packets>
```

```
          <rcv-packets>21</rcv-packets>
          <last-sent-seq>20</last-sent-seq>
          <last-rcv-seq>20</last-rcv-seq>
        </test-session>
      </session-reflector>
    </twamp>
  </data>
```

Appendix B. TWAMP Operational Commands

TWAMP operational commands could be performed programmatically or manually, e.g. using a command-line interface (CLI).

With respect to programmability, YANG can be used to define NETCONF Remote Procedure Calls (RPC), therefore it would be, in principle, possible to define TWAMP RPC operations for actions such as starting or stopping control connections or test sessions or groups of sessions; retrieving results; clearing stored results, and so on.

However, TWAMP [RFC5357] does not attempt to describe such operational actions. Refer also to Section 2 and the unlabeled links in Figure 1. In actual deployments different TWAMP implementations may support different sets of operational commands, with different restrictions. Therefore, this document considers it the responsibility of the individual implementation to define its corresponding TWAMP operational commands data model.

Authors' Addresses

Ruth Civil
Ciena Corporation
307 Legget Drive
Kanata, ON K2K 3C8
Canada

Email: gcivil@ciena.com
URI: www.ciena.com

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

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com

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

Email: rrahman@cisco.com

Mahesh Jethanandani
Xoriant Corporation
1248 Reamswood Drive
Sunnyvale, CA 94089
USA

Email: mjethanandani@gmail.com

Kostas Pentikousis (editor)
Travelping
Siemensdamm 50
Berlin 13629
Germany

Email: k.pentikousis@travelping.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2016

G. Mirsky
T. Elteto
Ericsson
March 7, 2016

Two-Way Active Measurement Protocol (TWAMP) Light Data Model
draft-mirsky-ippm-twamp-light-yang-02

Abstract

This document specifies the data model for implementations of Session-Sender and Session-Reflector for Two-Way Active Measurement Protocol (TWAMP) Light mode 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 September 8, 2016.

Table of Contents

1. Introduction

The Two-Way Active Measurement Protocol (TWAMP) [RFC5357] can be used to measure performance parameters of IP networks such as latency, jitter, and packet loss by sending test packets and monitoring their experience in the network. The [RFC5357] defines two protocols, TWAMP Control and TWAMP Test, and a profile of TWAMP Test, TWAMP Light. The TWAMP Light is known to have many implementations though no common management framework being defined, thus leaving some aspects of test packet processing to interpretation. The goal of this document is to collect analyze these variations; describe common model while allowing for extensions in the future. This document defines such a TWAMP data model and specifies it formally using the YANG data modeling language [RFC6020].

1.1. Conventions used in this document

1.1.1. Requirements Language

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

2. Scope, Model, and Applicability

The scope of this document includes model of the TWAMP Light as defined in Appendix I of [RFC5357] as well as model of accepted Errata. The former mode of TWAMP Light will be referred in this document as Stateless and the latter - Stateful. This document benefits from earlier attempt to define TWAMP MIB in [I-D.elteto-ippm-twamp-mib] and from TWAMP YANG model defined in [I-D.cmzrjp-ippm-twamp-yang].

Figure 1 updates TWAMP-Light reference model presented in Appendix I [RFC5357] for the scenario when instantiation of a TWAMP-Test session between Session-Sender and Session-Reflector controlled by communication between a Configuration Client as a manager and Configuration Servers as agents of the configuration session.

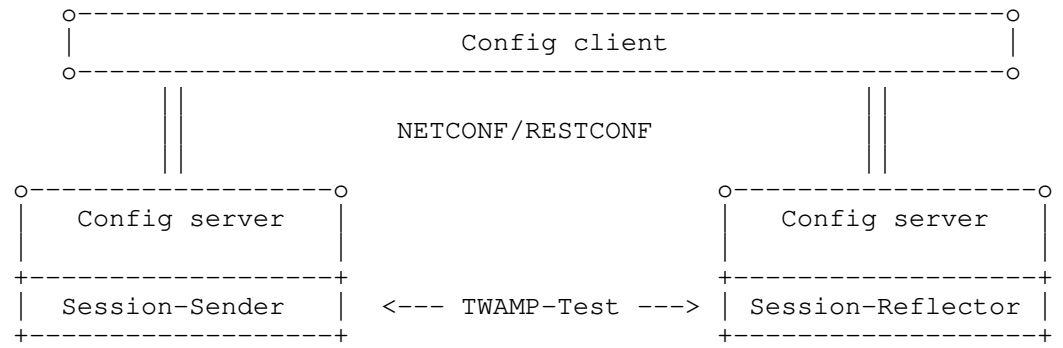


Figure 1: TWAMP Light Reference Model

2.1. Data Model Parameters

2.2. Session-Sender

TBA

2.3. Session-Reflector

TBA

3. Data Model

Creating TWAMP-Light data model presents number of challenges and among them is identification of a test-session at Session-Reflector. A Session-Reflector MAY require only as little as its IP and UDP port

number in received TWAMP-Test packet to spawn new test session. More so, to test processing of Class-of-Service along the same route in Equal Cost Multi-Path environment Session-Sender may run TWAMP test sessions concurrently using the same source IP address, source UDP port number, destination IP address, and destination UDP port number. Thus the only parameter that can be used to differentiate these test sessions would be DSCP value. The DSCP field may get re-marked along the path and without use of [RFC7750] that will go undetected, but by using five-tuple instead of four-tuple as a key we can ensure that TWAMP test packets that are considered as different test sessions follow the same path even in ECMP environments.

3.1. Tree Diagram

```
module: ietf-twamp-light
+--rw twampLightSessionSender {sessionSenderLight}?
|  +--rw testSession* [senderIp senderUdpPort reflectorIp
|                                 reflectorUdpPort dscp]
|     +--rw numberOfPackets?             uint32
|     +--rw packetPaddingSize?           uint32
|     +--rw sessionAuthenticationMode?   enumeration
|     +--rw interval?                    uint32
|     +--ro senderSessionState?          enumeration
|     +--ro sentPackets?                 uint32
|     +--ro rcvPackets?                  uint32
|     +--ro lastSentSeq?                 uint32
|     +--ro lastRcvSeq?                  uint32
|     +--rw senderIp                     inet:ip-address
|     +--rw senderUdpPort                inet:port-number
|     +--rw reflectorIp                  inet:ip-address
|     +--rw reflectorUdpPort             inet:port-number
|     +--rw dscp                         inet:dscp
+--rw twampLightSessionReflector {sessionReflectorLight}?
   +--rw reflectorLightState    boolean
   +--rw refwait?               uint32
   +--rw reflectorLightMode?    enumeration
   +--rw dscpHandlingMode?      enumeration
   +--rw testSession* [senderIp senderUdpPort reflectorIp
      |                          reflectorUdpPort dscp]
      +--ro sentPackets?        uint32
      +--ro rcvPackets?         uint32
      +--ro lastSentSeq?        uint32
      +--ro lastRcvSeq?         uint32
      +--rw senderIp            inet:ip-address
      +--rw senderUdpPort       inet:port-number
      +--rw reflectorIp         inet:ip-address
      +--rw reflectorUdpPort    inet:port-number
      +--rw dscp                inet:dscp
```

3.2. YANG Module

```
<CODE BEGINS> file "ietf-twamp-light@2016-0305"
module ietf-twamp-light {
   namespace "urn:ietf:params:xml:ns:yang:ietf-twamp-light";
    //namespace need to be assigned by IANA
   prefix "ietf-twamp-light";

   import ietf-inet-types {
     prefix inet;
   }
```

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

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

  description "TWAMP Light Data Model";

  revision "2016-03-05" {
    description "01 version. RFC5357 is covered,
including Appendix I and the Errata.";
    reference "draft-mirsky-ippm-twamp-light-yang";
  }

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

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

  grouping maintenanceStatistics {
    description "Maintenance statistics grouping";
    leaf sentPackets {
      type uint32;
      default 0;
      config "false";
      description "Packets sent";
    }
    leaf rcvPackets {
      type uint32;
      default 0;
      config "false";
      description "Packets received";
    }
    leaf lastSentSeq {
      type uint32;
      default 0;
      config "false";
      description "Last sent sequence number";
    }
    leaf lastRcvSeq {
      type uint32;
      default 0;
      config "false";
```

```
      description "Last received sequence number";
    }
  }

  grouping sessionLightParameters {
    description "Parameters common among Session-Sender and
Session-Reflector.";
    leaf senderIp {
      type inet:ip-address;
      description "Sender IP address";
    }
    leaf senderUdpPort {
      type inet:port-number {
          range "49152..65535";}
      description "Sender UDP port number";
    }
    leaf reflectorIp {
      type inet:ip-address;
      description "Reflector IP address";
    }
    leaf reflectorUdpPort {
      type inet:port-number{
          range "49152..65535";}
      description "Reflector UDP port number";
    }
    leaf dscp {
      type inet:dscp;
      description "The DSCP value to be placed in the header of TWAMP
          UDP
test packets generated by the Session-Sender. Whether
          Session-Reflector uses this value depends upon its local
configuration.";
    }
  }

   container twampLightSessionSender {
      if-feature sessionSenderLight;
      description "TWAMP-Light Session-Sender container";
      list testSession {
        key "senderIp senderUdpPort reflectorIp reflectorUdpPort dscp";
        ordered-by system;
        description "This structure is a container of test session
managed objects.";

        leaf numberOfPackets {
          type uint32;
          description "The overall number of UDP test packets to be
            transmitted by the sender for this test session.";
```

```
          }

          leaf packetPaddingSize {
            type uint32;
            default 27;
            description "Size of the Packet Padding. Suggested to run
            Path MTU Discovery
 to avoid packet fragmentation in IPv4
            and packet backholing in IPv6.";
          }

           leaf sessionAuthenticationMode {
             type enumeration {
               enum unathenticated {
                    description "Unauthenticated TWAMP-Light
                    test session";
               }
               enum authenticated {
                    description "Authenticated TWAMP-Light test session";
               }
               enum encrypted {
                    description "Encrypted TWAMP-Light test session";
               }
             }
             default unathenticated;
             description "Authentication mode of the TWAMP-Light test
             session."
;
           }

          leaf interval  {
            type uint32;
           description "Time interval between transmission of two
           consecutive packets in the test session.";
          }

          leaf senderSessionState {
            type enumeration {
              enum active {
                description "Test session is active.";
              }
              enum ready {
                description "Test session is idle.";
              }
            }
            default ready;
            config "false";
            description "State of the particular TWAMP-Light test
```

```
          session at the sender.";
        }
        uses maintenanceStatistics;
        uses sessionLightParameters;
      }
    }

    container twampLightSessionReflector {
      if-feature sessionReflectorLight;
      description "TWAMP-Light Session-Reflector container";
        leaf reflectorLightState {
          type boolean;
          mandatory "true";
          description "Whether this network element is enabled to
          act as TWAMP-Light Reflector";
        }

        leaf refwait {
        type uint32 {
          range 1..604800;
        }
        units seconds;
        default 900;
        description "REFWAIT(TWAMP test session timeout in seconds),
 the default value is 900";
      }

      leaf reflectorLightMode {
        type enumeration {
          enum stateful {
            description "When the Session-Reflector Light is stateful,
              i.e. is aware of test session state.";
          }
          enum stateless {
            description "when the Session-Reflector is statelss.";
          }
        }
        default stateless;
        description "Whether Session-Sender copies sequence number
          of received TWAMP-Test packet, i.e. Stateless, or counts
          reflected TWAMP-Test packets and restarts counter based on
          external event.";
       }

       leaf dscpHandlingMode {
         type enumeration {
           enum copyReceivedValue {
             description "Use DSCP value copied from received TWAMP
```

```
 test packet of the test session.";
              }
              enum useConfiguredValue {
                description "Use DSCP value configured for this test
 session on the Session-Reflector.";
              }
            }
            default copyReceivedValue;
            description "Session-Reflector handling of DSCP:
                - use value copied from received TWAMP-Test packet;
                - use value explicitly configured.";
          }

       list testSession {
         key "senderIp senderUdpPort reflectorIp reflectorUdpPort dscp";
         ordered-by system;
         description "This structure is a container of test session
 managed objects.";
         uses maintenanceStatistics;
         uses sessionLightParameters;
       }
     }
   }
 }
 <CODE ENDS>
```

4. 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-light

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-light

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

prefix: twamp

reference: RFC XXXX

5. Security Considerations

The configuration, state, action data defined in this document may be accessed via the NETCONF protocol [RFC6241]. SSH [RFC6242] is mandatory secure transport that is the lowest NETCONF layer. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

But, in general, this TWAMP Light YANG module does not change any underlying security issues that already may exist in [I-D.elteto-ippm-twamp-mib].

6. Acknowledgements

7. References

7.1. Normative References

[I-D.cmzrjp-ippm-twamp-yang]
Civil, R., Morton, A., Zheng, L., Rahman, R., Jethanandani, M., and K. Pentikousis, "Two-Way Active Measurement Protocol (TWAMP) Data Model", draft-cmzrjp-ippm-twamp-yang-02 (work in progress), October 2015.

[I-D.elteto-ippm-twamp-mib]
Elteto, T. and G. Mirsky, "Two-Way Active Measurement Protocol (TWAMP) Management Information Base (MIB)", draft-elteto-ippm-twamp-mib-01 (work in progress), January 2014.

[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>.

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

[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>.

[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>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

7.2. Informative References

[RFC7750] Hedin, J., Mirsky, G., and S. Baillargeon, "Differentiated Service Code Point and Explicit Congestion Notification Monitoring in the Two-Way Active Measurement Protocol (TWAMP)", RFC 7750, DOI 10.17487/RFC7750, February 2016, <http://www.rfc-editor.org/info/rfc7750>.

Authors' Addresses

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Tamas Elteto
Ericsson

Email: tamas.elteto@ericsson.com

Network Working Group
Internet-Draft
Updates: 2330 (if approved)
Intended status: Informational
Expires: June 13, 2016

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

December 11, 2015

IP Options and IPv6 Updates for IPPM's Active Metric Framework: Packets of Type-P and Standard-Formed Packets
draft-morton-ippm-2330-stdform-typep-02

Abstract

This memo updates the IP Performance Metrics (IPPM) Framework RFC 2330 with new considerations for measurement methodology and testing. The memo updates the definition of standard-formed packets in RFC 2330 to include IPv6 packets. It also augments distinguishing aspects of packets, referred to as Type-P for test packets in RFC 2330.

Two points (at least) are worthwhile discussing further: extent of coverage for 6LO and IPv6 Header Compression, and the continued need to define a "minimal standard-formed packet".

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 June 13, 2016.

Table of Contents

1. Introduction

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

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

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

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

2. Scope

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

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

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

3. Packets of Type-P

A fundamental property of many Internet metrics is that the measured value of the metric depends on characteristics of the IP packet(s) used to make the measurement. Potential influencing factors include IP header fields and their values, but also higher-layer protocol headers and their values. Consider an IP-connectivity metric: one obtains different results depending on whether one is interested in connectivity for packets destined for well-known TCP ports or unreserved UDP ports, or those with invalid IPv4 checksums, or those with TTL or Hop Limit of 16, for example. In some circumstances these distinctions will result in special treatment of packets in intermediate nodes and end systems (for example, if Diffserv [RFC2780], ECN [RFC3168], Router Alert, Hop-by-hop extensions [RFC7045], or Flow Labels [RFC6437] are used, or in the presence of firewalls or RSVP reservations).

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

fruitfully defined using generic Type-P definitions which are then made specific when performing actual measurements.

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

If the information constituting Type-P at the Source is found to have changed at the Destination (or at a measurement point between the Source and Destination, as in [RFC5644]), then the modified values SHOULD be noted and reported with the results. Some modifications occur according to the conditions encountered in transit (such as congestion notification) or due to the requirements of segments of the Source to Destination path. For example, the packet length will change if IP headers are converted to the alternate version/address family, or if optional Extension Headers are added or removed. Local policies in intermediate nodes based on examination of IPv6 Extension Headers may affect measurement repeatability. If intermediate nodes follow the recommendations of [RFC7045], repeatability may be improved to some degree.

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

A closely related note: it would be very useful to know if a given Internet component (like host, link, or path) treats equally a class C of different types of packets. If so, then any one of those types of packets can be used for subsequent measurement of the component. This suggests we devise a metric or suite of metrics that attempt to determine C.

4. Standard-Formed Packets

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

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

+ It is not an IP fragment.

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

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

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

o The version field is 4

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

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

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

o It does not contain IP options unless explicitly noted.

For an IPv6 ([RFC2460] and updates) packet to be standard-formed, the following criteria are required:

o The version field is 6.

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

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

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

- All parameters used in the header and Extension Headers are found in the IANA Registry of Internet Protocol Version 6 (IPv6) Parameters, partly specified in [RFC7045].

Compressed IPv6 headers must be compliant with [RFC4494], as updated by [RFC6282], in order to be declared "standard-formed".

The topic of IPv6 Extension Headers brings current controversies into focus as noted by [RFC6564] and [RFC7045]. The following additional considerations apply when IPv6 Extension Headers are present:

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

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

- Extension Header insertion or deletion: It is possible that Extension Headers could be added to, or removed from the header chain. The resulting packet may be standard-formed, with a corresponding Type-P.

- A change in packet length (from the corresponding packet observed at the Source) or header modification is a significant factor in Internet measurement, and requires a new Type-P to be reported.

We further require that if a packet is described as having a "length of B octets", then 0 <= B <= 65535; and if B is the payload length in octets, then B <= (65535-IP header size in octets, including any Extension Headers). The jumbograms defined in [RFC2675] are not covered by this length analysis. In practice, the path MTU will restrict the length of standard-formed packets that can successfully traverse the path.

So, for example, one might imagine defining an IP connectivity metric as "IP-type-P-connectivity for standard-formed packets with the IP Diffserv field set to 0", or, more succinctly, "IP-type-

P-connectivity with the IP Diffserv Field set to 0", since standard-formed is already implied by convention. Changing the contents of a field, such as the Diffserv Code Point, ECN bits, or Flow Label may have a profound affect on packet handling during transit, but does not affect a packet's status as standard-formed.

A particular type of standard-formed packet often useful to consider is the "minimal IP packet from A to B" - this is an IP packet with the following properties:

+ It is standard-formed.

+ Its data payload is 0 octets.

+ It contains no options or Extension Headers.

(Note that we do not define its protocol field, as different values may lead to different treatment by the network.)

When defining IP metrics we keep in mind that no packet smaller or simpler than this can be transmitted over a correctly operating IP network.

5. Conclusions

This memo adds the key considerations for utilizing IPv6 in two critical conventions of the IPPM Framework. It is RECOMMENDED to adopt these new considerations in measurements involving IPv6.

6. Security Considerations

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

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

7. IANA Considerations

This memo makes no requests of IANA.

8. Acknowledgements

The authors thank Brian Carpenter for identifying the lack of IPv6 coverage in IPPM's Framework, and for listing additional distinguishing factors for packets of Type-P. Both Brian and Fred Baker discussed many of the interesting aspects of IPv6 with the co-authors, leading to a more solid first draft: thank you both. Thanks to Bill Jouris for an editorial pass through the pre-00 text.

9. References

9.1. Normative References

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

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

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

[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>.

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

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, DOI 10.17487/RFC2780, March 2000, <http://www.rfc-editor.org/info/rfc2780>.

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

[RFC4494] Song, JH., Poovendran, R., and J. Lee, "The AES-CMAC-96 Algorithm and Its Use with IPsec", RFC 4494, DOI 10.17487/RFC4494, June 2006, <http://www.rfc-editor.org/info/rfc4494>.

[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>.

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

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

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

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

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

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

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

9.2. Informative References

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

Authors' Addresses

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

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

Joachim Fabini
TU Wien
Gusshausstrasse 25/E389
Vienna 1040
Austria

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

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

Email: nalini.elkins@insidethestack.com

Michael S. Ackermann
Blue Cross Blue Shield of Michigan

Email: mackermann@bcbsm.com

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 24, 2016

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

February 21, 2016

Initial Performance Metric Registry Entries
draft-morton-ippm-initial-registry-04

Abstract

This memo defines the Initial Entries for the Performance Metrics Registry.

Version 04 * All section 4 parameters reference YANG types for alternate data formats. * Discussion has concluded that usecase(s) for machine parse-able registry columns are not needed.

Still need: * suggestion of standard naming format for parameters. * revisions that follow section 4 changes in other proposed metrics.

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

Table of Contents

1. Introduction

Note: Efforts to synchronize structure and terminology with [I-D.ietf-ippm-metric-registry] will likely be incomplete until both drafts are stable.

This memo proposes an initial set of entries for the Performance Metric Registry. It uses terms and definitions from the IPPM literature, primarily [RFC2330]. Proponents of Passive Performance Metrics are encouraged to develop a similar document.

Although there are several standard templates for organizing specifications of performance metrics (see [RFC2679] for an example of the traditional IPPM template, based to large extent on the Benchmarking Methodology Working Group's traditional template in [RFC1242], and see [RFC6390] for a similar template), none of these templates were intended to become the basis for the columns of an IETF-wide registry of metrics. While examinating aspects of metric specifications which need to be registered, it became clear that none of the existing metric templates fully satisfies the particular needs of a registry.

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

2. Scope

This document defines the initial set of Performance Metrics Registry entries, for which IETF approval (following development in the IP Performance Metrics (IPPM) Working Group) will satisfy the requirement for Expert Review. Note that all are Active Performance Metrics, which are based on RFCs prepared in the IPPM working group of the IETF, according to their framework [RFC2330] and its updates.

3. Registry Categories and Columns

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

Registry Categories and Columns, shown as

	Category	
	Column	Column

Summary

ID	Name	URIs	Description

Metric Definition

Reference Definition	Fixed Parameters

Method of Measurement

Reference Method	Packet Generation Stream	Traffic Filter	Sampling dist.	Run-time Param	Role

Output

Type	Reference Definition	Units

Administrative information

Status	Request	Rev	Rev.Date

Comments and Remarks

4. UDP Round-trip Latency Registry Entry

This section gives an initial registry entry for the UDP Round-trip Latency.

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

4.1. Summary

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

4.1.1. ID (Identifier)

<insert a numeric identifier, an integer, TBD>

4.1.2. Name

<insert name according to metric naming convention>

Act_IP_UDP_Round-trip_Delay_Poisson_95th-percentile

4.1.3. URIs

URN: Prefix urn:ietf:params:performance:metric...<name>

URL: http://<TBD by IANA>/<name>

4.1.4. Description

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

4.2. Metric Definition

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

4.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

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

[RFC2681]

<specific section reference and additional clarifications, if needed>

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

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

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

4.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL: set to 255
  - Protocol: Set to 17 (UDP)
- IPv6 header values:
  - DSCP: set to 0
  - Hop Count: set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum MUST be calculated

- UDP Payload
  - total of 9 bytes

Other measurement parameters:

- Tmax: a loss threshold waiting time
  - 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 5 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

4.3. Method of Measurement

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

4.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

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

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

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

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

packet. Sequence numbers or other send-order identification MUST be retained at the Src or included with each packet to dis-ambiguate packet reordering if it occurs.

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

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

4.3.2. Packet Generation Stream

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

<section/specification references, and description of any new generation parameters, if needed>

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

>>> Check with Sam, most likely it is this...

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

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

4.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

<section reference>.

NA

4.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

NA

4.3.5. Run-time Parameters and Data Format

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

<list of run-time parameters, and their data formats>

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

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

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

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

Reciprocal_lambda average packet interval for Poisson Streams expressed in units of seconds, as a positive value of type

decimal64 with fraction digits = 5 (see section 9.3 of [RFC6020]) with resolution of 0.0001 seconds (0.1 ms), and with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

Trunc Upper limit on Poisson distribution expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 5 (see section 9.3 of [RFC6020]) with resolution of 0.0001 seconds (0.1 ms), and with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905] (values above this limit will be clipped and set to the limit value). (if fixed, Trunc = 30.0000 seconds.)

>>> should Poisson run-time params be fixed instead? probably yes if modeling a specific version of MBA tests.

4.3.6. Roles

<lists the names of the different roles from the measurement method>

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

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

4.4. Output

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

4.4.1. Type

<insert name of the output type, raw or a selected summary statistic>

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

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

The percentile = 95, meaning that the reported delay, "Percentile95", is the smallest value of Round-trip delay for which the Empirical Distribution Function (EDF), F(Percentile95) >= 95% of the singleton Round-trip delay values in the conditional distribution. See section 11.3 of [RFC2330] for the definition of the percentile statistic using the EDF.

4.4.2. Data Format

<describe the data format for each type of result>

For all outputs ---

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

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

Raw -- REMOVED IN VERSION 01

For Act_IP_UDP_Round-trip_Delay_Poisson_95th-percentile:

Percentile95 The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

4.4.3. Reference

<pointer to section/spec where output type/format is defined>

See the Data Format column for references.

4.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

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

4.5. Administrative items

4.5.1. Status

<current or deprecated>

4.5.2. Requestor (keep?)

name or RFC, etc.

4.5.3. Revision

1.0

4.5.4. Revision Date

YYYY-MM-DD

4.6. Comments and Remarks

Additional (Informational) details for this entry

5. Packet Delay Variation Registry Entry

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

Note: If each Registry entry should only produce a "raw" output or a statistical summary, then the "Output" Category can be split and this section can become two closely-related metrics.

5.1. Summary

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

<skipping some Summary columns for now>

5.1.1. ID (Identifier)

<insert numeric identifier, an integer>

5.1.2. Name

<insert name according to metric naming convention>

Act_IP-UDP-One-way-pdv-95th-percentile-Poisson

URL: ??

5.1.3. URI

URI: Prefix urn:ietf:params:performance:metric<add name>

5.1.4. Description

An assessment of packet delay variation with respect to the minimum delay observed on the stream.

5.2. Metric Definition

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

5.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

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

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

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

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

<specific section reference and additional clarifications, if needed>

See sections 2.4 and 3.4 of [RFC3393]. Singleton delay differences measured are referred to by the variable name "ddT".

5.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

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

- L, a packet length in bits. L = 200 bits.

- Tmax, a maximum waiting time for packets to arrive at Dst, set sufficiently long to disambiguate packets with long delays from packets that are discarded (lost). Tmax = 3 seconds.

- Type-P, as defined in [RFC2330], which includes any field that may affect a packet's treatment as it traverses the network. The packets are IP/UDP, with DSCP = 0 (BE).

5.3. Method of Measurement

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

5.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

See section 2.6 and 3.6 of [RFC3393] for singleton elements.

5.3.2. Packet Generation Stream

<list of generation parameters and section/spec references if needed>

Poisson distributed as described in [RFC2330], with the following Parameters.

- lambda, a rate in reciprocal seconds (for Poisson Streams). lambda = 1 packet per second

- Upper limit on Poisson distribution (values above this limit will be clipped and set to the limit value). Upper limit = 30 seconds.

5.3.3. Traffic Filtering (observation) Details

<insert the measured results based on a filtered version of the packets observed, and this section provides the filter details (when present), and section reference>.

NA

5.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

NA

5.3.5. Run-time Parameters and Data Format

<list of run-time parameters, and any reference(s)>.

- Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- T, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]). When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval.

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]), interpreted as the Duration of the measurement interval.

5.3.6. Roles

<lists the names of the different roles from the measurement method>

Src - the host that sends the stream of packets.

Dst - the host that receives the stream of packets.

5.4. Output

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

5.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

Raw -- for each packet sent, pairs of values.

Percentile -- for the conditional distribution of all packets with a valid value of one-way delay (undefined delays are excluded), a

single value corresponding to the 95th percentile of the singletons, ddT.

5.4.2. Data Format

<describe the data format for each type of result>

For all Output types

- T, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

Raw -

- T1, the wire time of the first packet in a pair, measured at MP(Src) as it leaves for Dst (64-bit NTP Timestamp Format, see section 6 of [RFC5905]).

- T2, the wire time of the second packet in a pair, measured at MP(Src) as it leaves for Dst (64-bit NTP Timestamp Format, see section 6 of [RFC5905]).

- I(i),I(i+1), i >=0, pairs of times which mark the beginning and ending of the intervals in which the packet stream from which the measurement is taken occurs. Here, I(0) = T0 and assuming that n is the largest index, I(n) = Tf (pairs of 64-bit NTP Timestamp Format, see section 6 of [RFC5905]).

- When the one-way delay of a packet in the calculation pair for ddT is undefined, then ddT is undefined for that pair.

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

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

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

The percentile = 95.

Data format is a 32-bit signed floating point value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

5.4.3. Reference

<pointer to section/spec where output type/format is defined>

see Data Format column.

5.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

See section 3.3 of [RFC3393] for singleton elements, ddT. The units are seconds, and the same units are used for 95th percentile.

[RFC2330] recommends that when a time is given, it will be expressed in UTC.

The timestamp format (for T, Tf, etc.) is the same as in [RFC5905] (64 bits) and is as follows: the first 32 bits represent the unsigned integer number of seconds elapsed since 0h on 1 January 1900; the next 32 bits represent the fractional part of a second that has elapsed since then.

5.5. Administrative items

5.5.1. Status

<current or depricated>

5.5.2. Requestor (keep?)

<name of individual or RFC, etc.>

5.5.3. Revision

1.0

5.5.4. Revision Date

YYYY-MM-DD

5.6. Comments and Remarks

<Additional (Informational) details for this entry>

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

6. DNS Response Latency Registry Entry

This section gives an initial registry entry for DNS Response Latency. RFC 2681 [RFC2681] defines a Round-trip delay metric. We build on that metric by specifying several of the input parameters to precisely define a metric for measuring DNS latency.

6.1. Summary

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

<skipping some admin columns for now>

6.1.1. ID (Identifier)

<insert numeric identifier, an integer>

6.1.2. Name

<insert name according to metric naming convention>

URL: ??

6.1.3. URI

URI: Prefix urn:ietf:params:performance:metric

6.1.4. Description

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

6.2. Metric Definition

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

6.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

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

[RFC1035]

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

[RFC2681]

<specific section reference and additional clarifications, if needed>

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

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

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

6.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255

- Protocol: Set to 17 (UDP)

- UDP header values:
  - Source port: 53
  - Destination port: 53
  - Checksum: the checksum must be calculated

- Payload: The payload contains a DNS message as defined in RFC 1035 [RFC1035] with the following values:
  - The DNS header section contains:
    - QR: set to 0 (Query)
    - OPCODE: set to 0 (standard query)
    - AA: not set
    - TC: not set
    - RD: set to one (recursion desired)
    - RA: not set
    - RCODE: not set
    - QDCOUNT: set to one (only one entry)
    - ANCOUNT: not set
    - NSCOUNT: not set
    - ARCOUNT: not set
  - The Question section contains:
    - QNAME: the FQDN provided as input for the test
    - QTYPE: the query type provided as input for the test
    - QCLASS: set to IN
  - The other sections do not contain any Resource Records.

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

Timeout: Tmax = 5 seconds (to help disambiguate queries)

6.3. Method of Measurement

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

6.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

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

The method requires sequence numbers or other send-order information to be retained at the Src or included with each packet to dis-ambiguate packet reordering if it occurs. Sequence number is part of the payload described under Fixed Parameters.

DNS Messages bearing Queries provide for random ID Numbers, so more than one query may be launched while a previous request is outstanding when the ID Number is used.

IF a DNS response does not arrive within Tmax, the result is undefined. The Message ID SHALL be used to disambiguate the successive queries.

>>> This would require support of ID generation and population in the Message. An alternative would be to use a random Source port on the Query Message, but we would choose ONE before proceding.

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

6.3.2. Packet Generation Stream

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

<list of generation parameters and section/spec references if needed>

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

>>> Check with Sam, most likely it is this...

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

6.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

<section reference>.

NA

6.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

NA

6.3.5. Run-time Parameters and Data Format

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

<list of run-time parameters, and their data formats>

- Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]). When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval.

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]), interpreted as the Duration of the measurement interval.

- 1/lambda, average packet rate (for Poisson Streams). (1/lambda = 0.1 packet per second, if fixed)

- Upper limit on Poisson distribution (values above this limit will be clipped and set to the limit value). (if fixed, Upper limit = 300 seconds.)

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

The format for 1/lambda and Upper limit of Poisson Dist. are the short format in [RFC5905] (32 bits) and is as follows: the first 16 bits represent the integer number of seconds; the next 16 bits represent the fractional part of a second.

>>> should Poisson run-time params be fixed instead? probably yes if modeling a specific version of MBA tests.

6.3.6. Roles

<lists the names of the different roles from the measurement method>

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

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

6.4. Output

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

6.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

For all output types:

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

Raw -- for each packet sent, pairs of values.

>>> and the status of the response, only assigning values to successful query-response pairs.

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

6.4.2. Data Format

<describe the data format for each type of result>

Raw -- for each packet sent, pairs of values as follows:

- T, the time when the packet was sent from Src, 128-bit NTP Date Format, see section 6 of [RFC5905])

- dT, a value of Round-trip delay, format is *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

- dT is undefined when the packet is not received at Src in waiting time Tmxax seconds (need undefined code for no-response or un-successful response)

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

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

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

The percentile = 95.

Data format is a 32-bit signed floating point value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

6.4.3. Reference

<pointer to section/spec where output type/format is defined>

See the Data Format column for references.

6.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

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

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

6.5. Administrative items

6.5.1. Status

<current or depricated>

6.5.2. Requestor (keep?)

name or RFC, etc.

6.5.3. Revision

1.0

6.5.4. Revision Date

YYYY-MM-DD

6.6. Comments and Remarks

Additional (Informational) details for this entry

7. UDP Poisson One-way Delay Registry Entries

This section gives an initial registry entry for the UDP Poisson One-way Delay.

Note: Each Registry "Name" below specifies a single registry entry, whose output format varies according to a component of the name that specifies one form of statistical summary.

IANA is asked to assign a different numeric identifiers to each Name. All column entries beside the Summary and Output categories are the same, thus this section proposes five closely-related registry entries. As a result, IANA is also asked to assign corresponding URIs and URLs.

7.1. Summary

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

7.1.1. ID (Identifier)

<insert numeric identifier, an integer, one corresponding to each name below>

7.1.2. Name

<insert name according to metric naming convention>

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_<statistic>

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Percentile95

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Mean

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Min

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Max

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Std_Dev

7.1.3. URI and URL

URI: Prefix urn:ietf:params:performance:metric...<name>

URL: http:\\www.iana.org\ ... <name>

7.1.4. Description

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

7.2. Metric Definition

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

7.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

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

[RFC2679]

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

[RFC6049]

<specific section reference and additional clarifications, if needed>

Section 3.4 of [RFC2679] provides the reference definition of the singleton (single value) One-way delay metric. Section 4.4 of [RFC2679] provides the reference definition expanded to cover a multi-value sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

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

NOTE: RFC2679 will be replaced by 2679-bis on approval, see draft-ietf-ippm-2679-bis-01.

7.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum must be calculated
- UDP Payload: TWAMP Test Packet Formats, Section 4.1.2 of [RFC5357]
  - Security features in use influence the number of Padding octets.
  - 250 octets total, including the TWAMP format

Timeout, Tmax: 3 seconds

7.3. Method of Measurement

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

7.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

The methodology for this metric is defined as Type-P-One-way-Delay-Poisson-Stream in section 3.6 of [RFC2679] and section 4.6 of [RFC2679] using the Type-P and Timeout defined under Fixed Parameters.

The method requires sequence numbers or other send-order information to be retained at the Src or included with each packet to dis-ambiguate packet reordering if it occurs. Sequence number is part of the TWAMP payload described under Fixed Parameters.

7.3.2. Packet Generation Stream

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

<list of generation parameters and section/spec references if needed>

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

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

7.3.3. Traffic Filtering (observation) Details

NA

7.3.4. Sampling Distribution

NA

7.3.5. Run-time Parameters and Data Format

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

<list of run-time parameters, and their data formats>

- Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]). When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval.

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]), interpreted as the Duration of the measurement interval.

- 1/lambda, average packet rate (for Poisson Streams). (1/lambda = 1 packet per second, if fixed)

- Upper limit on Poisson distribution (values above this limit will be clipped and set to the limit value). (if fixed, Upper limit = 30 seconds.)

The format for 1/lambda and Upper limit of Poisson Dist. are the short format in [RFC5905] (32 bits) and is as follows: the first 16 bits represent the integer number of seconds; the next 16 bits represent the fractional part of a second.

>>> should Poisson run-time params be fixed instead? probably yes if modeling a specific version of tests. Note in the NAME, i.e. Poisson3.3

7.3.6. Roles

<lists the names of the different roles from the measurement method>

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

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

7.4. Output

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

7.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

See subsection titles below for Types.

7.4.2. Data Format

<describe the data format for each type of result>

For all output types ---

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

7.4.2.1. Percentile95

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

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

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

The percentile = 95.

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

7.4.2.2. Mean

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

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

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

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

7.4.2.3. Min

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

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

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

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

7.4.2.4. Max

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

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

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

```
Max = (FiniteDelay [j])

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

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

7.4.2.5. Std_Dev

7.4.3. Reference

<pointer to section/spec where output type/format is defined>

See the Data Format column for references.

7.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

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

The 95th Percentile of One-way Delay is expressed in seconds.

7.5. Administrative items

7.5.1. Status

<current or depricated>

7.5.2. Requestor (keep?)

name or RFC, etc.

7.5.3. Revision

1.0

7.5.4. Revision Date

YYYY-MM-DD

7.6. Comments and Remarks

Additional (Informational) details for this entry

8. UDP Periodic One-way Delay Registry Entries

This section gives an initial registry entry for the UDP Periodic One-way Delay.

Note: Each Registry "Name" below specifies a single registry entry, whose output format varies according to a component of the name that specifies one form of statistical summary.

IANA is asked to assign a different numeric identifiers to each Name. All other column entries are the same, thus this section is proposes five closely-related registry entries. As a result, IANA is also asked to assign corresponding URIs and URLs.

8.1. Summary

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

8.1.1. ID (Identifier)

<insert numeric identifier, an integer, one corresponding to each name below>

8.1.2. Name

<insert name according to metric naming convention>

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_<statistic>

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Percentile95

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Mean

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Min

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Max

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Std_Dev

8.1.3. URI and URL

URI: Prefix urn:ietf:params:performance:metric...<name>

URL: http:\\www.iana.org\ ... <name>

8.1.4. Description

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

8.2. Metric Definition

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

8.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

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

[RFC2679]

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

[RFC6049]

<specific section reference and additional clarifications, if needed>

Section 3.4 of [RFC2679] provides the reference definition of the singleton (single value) One-way delay metric. Section 4.4 of [RFC2679] provides the reference definition expanded to cover a multi-value sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

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

NOTE: RFC2679 will be replaced by 2679-bis on approval, see draft-ietf-ippm-2679-bis-01.

ANY other conditions, ...

8.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum must be calculated
- UDP Payload: TWAMP Test Packet Formats, Section 4.1.2 of [RFC5357]
  - Security features in use influence the number of Padding octets.
  - 142 octets total, including the TWAMP format

Timeout, Tmax: 3 seconds

8.3. Method of Measurement

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

8.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

The methodology for this metric is defined as Type-P-One-way-Delay-Poisson-Stream in section 3.6 of [RFC2679] and section 4.6 of [RFC2679] using the Type-P and Timeout defined under Fixed Parameters.

The method requires sequence numbers or other send-order information to be retained at the Src or included with each packet to dis-ambiguate packet reordering if it occurs. Sequence number is part of the TWAMP payload described under Fixed Parameters.

8.3.2. Packet Generation Stream

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

<list of generation parameters and section/spec references if needed>

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

- incT, the nominal duration of inter-packet interval, first bit to first bit
- dT, the duration of the interval for allowed sample start times
- T0, the actual start time

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

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

8.3.3. Traffic Filtering (observation) Details

NA

8.3.4. Sampling Distribution

NA

8.3.5. Run-time Parameters and Data Format

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

<list of run-time parameters, and their data formats>

- Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]). When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval.

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]), interpreted as the Duration of the measurement interval.

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

- dT, the duration of the interval for allowed sample start times

The format for incT and dT are the short format in [RFC5905] (32 bits) and is as follows: the first 16 bits represent the integer number of seconds; the next 16 bits represent the fractional part of a second.

>>> should Periodic run-time params be fixed instead? probably yes if modeling a specific version of tests. Note in the NAME, i.e. Poisson3.3

8.3.6. Roles

<lists the names of the different roles from the measurement method>

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

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

8.4. Output

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

8.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

See subsection titles in Data Format for Types.

8.4.2. Data Format

<describe the data format for each type of result>

For all output types ---

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

8.4.2.1. Percentile95

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

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

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

The percentile = 95.

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

8.4.2.2. Mean

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

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

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

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

8.4.2.3. Min

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

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

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

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

8.4.2.4. Max

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

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

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

```
Max = (FiniteDelay [j])

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

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

8.4.2.5. Std_Dev

8.4.3. Reference

<pointer to section/spec where output type/format is defined>

See the Data Format column for references.

8.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

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

8.5. Administrative items

8.5.1. Status

<current or depricated>

8.5.2. Requestor (keep?)

name or RFC, etc.

8.5.3. Revision

1.0

8.5.4. Revision Date

YYYY-MM-DD

8.6. Comments and Remarks

Additional (Informational) details for this entry

9. partly BLANK Registry Entry

This section gives an initial registry entry for ....

9.1. Summary

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

<skipping the admin columns for now>

9.1.1. ID (Identifier)

<insert numeric identifier, an integer>

9.1.2. Name

<insert name according to metric naming convention>

URL: ??

9.1.3. URI

URI: Prefix urn:ietf:params:performance:metric

9.1.4. Description

TBD.

9.2. Metric Definition

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

9.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

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

<specific section reference and additional clarifications, if needed>

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

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

<<< Check how the Methodology also makes this clear (or not) >>>

9.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum must be calculated
- Payload
  - Sequence number: 8-byte integer
  - Timestamp: 8 byte integer. Expressed as 64-bit NTP timestamp as per section 6 of RFC 5905 [RFC5905]
  - No padding (total of 9 bytes)

Timeout: 3 seconds

9.3. Method of Measurement

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

9.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

9.3.2. Packet Generation Stream

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

<list of generation parameters and section/spec references if needed>

9.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

<section reference>.

9.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

9.3.5. Run-time Parameters and Data Format

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

<list of run-time parameters>

<reference(s)>.

9.3.6. Roles

<lists the names of the different roles from the measurement method>

9.4. Output

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

9.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

9.4.2. Data Format

<describe the data format for each type of result>

- Value:
- Data Format: (There may be some precedent to follow here, but otherwise use 64-bit NTP Timestamp Format, see section 6 of [RFC5905]).
- Reference: <section reference>

9.4.3. Reference

<pointer to section/spec where output type/format is defined>

9.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

9.5. Administrative items

9.5.1. Status

<current or depricated>

9.5.2. Requestor (keep?)

name or RFC, etc.

9.5.3. Revision

1.0

9.5.4. Revision Date

YYYY-MM-DD

9.6. Comments and Remarks

Additional (Informational) details for this entry

10. BLANK Registry Entry

This section gives an initial registry entry for ....

10.1. Summary

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

<skipping the Summary columns for now>

10.1.1. ID (Identifier)

<insert numeric identifier, an integer>

10.1.2. Name

<insert name according to metric naming convention>

URL: ??

10.1.3. URI

URI: Prefix urn:ietf:params:performance:metric

10.1.4. Description

TBD.

10.2. Metric Definition

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

10.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

<specific section reference and additional clarifications, if needed>

10.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

10.3. Method of Measurement

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

10.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

10.3.2. Packet Generation Stream

<list of generation parameters and section/spec references if needed>

10.3.3. Traffic Filtering (observation) Details

<insert the measured results based on a filtered version of the packets observed, and this section provides the filter details (when present), and section reference>.

10.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

10.3.5. Run-time Parameters and Data Format

<list of run-time parameters, and any reference(s)>.

10.3.6. Roles

<lists the names of the different roles from the measurement method>

10.4. Output

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

10.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

10.4.2. Data Format

<describe the data format for each type of result>

10.4.3. Reference

<pointer to section/spec where output type/format is defined>

10.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

10.5. Administrative items

10.5.1. Status

<current or depricated>

10.5.2. Requestor (keep?)

<name of individual or RFC, etc.>

10.5.3. Revision

1.0

10.5.4. Revision Date

YYYY-MM-DD

10.6. Comments and Remarks

Additional (Informational) details for this entry

11. Example RTCP-XR Registry Entry

This section is MAY BE DELETED or adapted before submission.

This section gives an example registry entry for the end-point metric described in RFC 7003 [RFC7003], for RTCP-XR Burst/Gap Discard Metric reporting.

11.1. Registry Indexes

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

11.1.1. Identifier

An integer having enough digits to uniquely identify each entry in the Registry.

11.1.2. Name

A metric naming convention is TBD.

11.1.3. URI

Prefix urn:ietf:params:performance:metric

11.1.4. Status

current

11.1.5. Requestor

Alcelip Mornuley

11.1.6. Revision

1.0

11.1.7. Revision Date

2014-07-04

11.1.8. Description

TBD.

11.1.9. Reference Specification(s)

[RFC3611][RFC4566][RFC6776][RFC6792][RFC7003]

11.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters. Section 3.2 of [RFC7003] provides the reference information for this category.

11.2.1. Reference Definition

Packets Discarded in Bursts:

The total number of packets discarded during discard bursts. The measured value is unsigned value. If the measured value exceeds 0xFFFFFD, the value 0xFFFFFE MUST be reported to indicate an over-range measurement. If the measurement is unavailable, the value 0xFFFFFF MUST be reported.

11.2.2. Fixed Parameters

Fixed Parameters are input factors that must be determined and embedded in the measurement system for use when needed. The values of these parameters is specified in the Registry.

Threshold: 8 bits, set to value = 3 packets.

The Threshold is equivalent to Gmin in [RFC3611], i.e., the number of successive packets that must not be discarded prior to and following a discard packet in order for this discarded packet to be regarded as part of a gap. Note that the Threshold is set in accordance with the Gmin calculation defined in Section 4.7.2 of [RFC3611].

Interval Metric flag: 2 bits, set to value 11=Cumulative Duration

This field is used to indicate whether the burst/gap discard metrics are Sampled, Interval, or Cumulative metrics [RFC6792]:

I=10: Interval Duration - the reported value applies to the most recent measurement interval duration between successive metrics reports.

I=11: Cumulative Duration - the reported value applies to the accumulation period characteristic of cumulative measurements.

Senders MUST NOT use the values I=00 or I=01.

11.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations. For the Burst/Gap Discard Metric, it appears that the only guidance on methods of measurement is in Section 3.0 of [RFC7003] and its supporting references. Relevant information is repeated below, although there appears to be no section titled "Method of Measurement" in [RFC7003].

11.3.1. Reference Method

Metrics in this block report on burst/gap discard in the stream arriving at the RTP system. Measurements of these metrics are made at the receiving end of the RTP stream. Instances of this metrics block use the synchronization source (SSRC) to refer to the separate auxiliary Measurement Information Block [RFC6776], which describes measurement periods in use (see [RFC6776], Section 4.2).

This metrics block relies on the measurement period in the Measurement Information Block indicating the span of the report. Senders MUST send this block in the same compound RTCP packet as the Measurement Information Block. Receivers MUST verify that the measurement period is received in the same compound RTCP packet as this metrics block. If not, this metrics block MUST be discarded.

11.3.2. Stream Type and Stream Parameters

Since RTCP-XR Measurements are conducted on live RTP traffic, the complete description of the stream is contained in SDP messages that proceed the establishment of a compatible stream between two or more communicating hosts. See Run-time Parameters, below.

11.3.3. Output Type and Data Format

The output type defines the type of result that the metric produces.

- Value: Packets Discarded in Bursts
- Data Format: 24 bits
- Reference: Section 3.2 of [RFC7003]

11.3.4. Metric Units

The measured results are apparently expressed in packets, although there is no section of [RFC7003] titled "Metric Units".

11.3.5. Run-time Parameters and Data Format

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

The Data Format of each Run-time Parameter SHALL be specified in this column, to simplify the control and implementation of measurement devices.

SSRC of Source: 32 bits As defined in Section 4.1 of [RFC3611].

SDP Parameters: As defined in [RFC4566]

Session description v= (protocol version number, currently only 0)

o= (originator and session identifier : username, id, version number, network address)

s= (session name : mandatory with at least one UTF-8-encoded character)

i=* (session title or short information) u=* (URI of description)

e=* (zero or more email address with optional name of contacts)

p=* (zero or more phone number with optional name of contacts)

c=* (connection information--not required if included in all media)

b=* (zero or more bandwidth information lines) One or more Time descriptions ("t=" and "r=" lines; see below)

z=* (time zone adjustments)

k=* (encryption key)

a=* (zero or more session attribute lines)

Zero or more Media descriptions (each one starting by an "m=" line; see below)

m= (media name and transport address)

i=* (media title or information field)

c=* (connection information -- optional if included at session level)

b=* (zero or more bandwidth information lines)

k=* (encryption key)

a=* (zero or more media attribute lines -- overriding the Session attribute lines)

An example Run-time SDP description follows:

v=0

o=jdoe 2890844526 2890842807 IN IP4 192.0.2.5

s=SDP Seminar i=A Seminar on the session description protocol

u=http://www.example.com/seminars/sdp.pdf e=j.doe@example.com (Jane Doe)

c=IN IP4 233.252.0.12/127

t=2873397496 2873404696

a=recvonly

m=audio 49170 RTP/AVP 0

m=video 51372 RTP/AVP 99

a=rtpmap:99 h263-1998/90000

11.4. Comments and Remarks

TBD.

12. Revision History

This section may be removed for publication. It contains partial information on updtes.

This draft replaced draft-mornuley-ippm-initial-registry.

In version 02, Section 4 has been edited to reflect recent discussion on the ippm-list: * Removed the combination or "Raw" and left 95th percentile. * Hanging Indent on Run-time parameters (Fixed parameters use bullet lists and other indenting formats. * Payload format for

measurement has been removed. * Explanation of Conditional delay distribution.

Version 03 addressed Phil Eardley's comments and suggestions in sections 1-4. and resolved the definition of Percentiles.

Version 04 * All section 4 parameters reference YANG types for alternate data formats. * Discussion has concluded that usecase(s) for machine parse-able registry columns are not needed.

Still need: * suggestion of standard naming format for parameters.

Note: lambda parameter description is correct in section 4, elsewhere needs fix.

13. Security Considerations

These registry entries represent no known security implications for Internet Security. Each referenced Metric contains a Security Considerations section.

14. IANA Considerations

IANA is requested to populate The Performance Metric Registry defined in [I-D.ietf-ippm-metric-registry] with the values defined above.

<more is needed here>

15. Acknowledgements

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

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

16. References

16.1. Normative References

[I-D.ietf-ippm-metric-registry]
Bagnulo, M., Claise, B., Eardley, P., and A. Morton, "Registry for Performance Metrics", Internet Draft (work in progress) draft-ietf-ippm-metric-registry, 2014.

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

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

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

[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>.

[RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, DOI 10.17487/RFC2680, September 1999, <http://www.rfc-editor.org/info/rfc2680>.

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

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

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

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

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

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

[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>.

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

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

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

16.2. Informative References

[Brow00] Brownlee, N., "Packet Matching for NeTraMet Distributions", March 2000.

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

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <http://www.rfc-editor.org/info/rfc3611>.

[RFC4148] Stephan, E., "IP Performance Metrics (IPPM) Metrics Registry", BCP 108, RFC 4148, DOI 10.17487/RFC4148, August 2005, <http://www.rfc-editor.org/info/rfc4148>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC 5472, DOI 10.17487/RFC5472, March 2009, <http://www.rfc-editor.org/info/rfc5472>.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, DOI 10.17487/RFC5477, March 2009, <http://www.rfc-editor.org/info/rfc5477>.

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

[RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, DOI 10.17487/RFC6248, April 2011, <http://www.rfc-editor.org/info/rfc6248>.

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

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

[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, DOI 10.17487/RFC6776, October 2012, <http://www.rfc-editor.org/info/rfc6776>.

[RFC6792] Wu, Q., Ed., Hunt, G., and P. Arden, "Guidelines for Use of the RTP Monitoring Framework", RFC 6792, DOI 10.17487/RFC6792, November 2012, <http://www.rfc-editor.org/info/rfc6792>.

[RFC7003] Clark, A., Huang, R., and Q. Wu, Ed., "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting", RFC 7003, DOI 10.17487/RFC7003, September 2013, <http://www.rfc-editor.org/info/rfc7003>.

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

Authors' Addresses

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

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

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich
ENGLAND

Email: philip.eardley@bt.com

Kevin D'Souza
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 xxxx
Email: kld@att.com

IP Performance Metrics Group
Internet-Draft
Intended status: Informational
Expires: April 7, 2016

Srivathsa Sarangapani
Peyush Gupta
Juniper Networks
V. Hegde
Consultant

October 5, 2015

Monitoring Service KPIs using TWAMP - Methodology
draft-spv-ippm-monitor-methodology-services-kpi-00

Abstract

We are introducing a new method to calculate services KPIs and metrics in the network using TWAMP protocol. The services here ranging from subscriber aware services to security application, Traffic load balancing, content delivery, real time streaming and like.The KPIs discussed in this draft include Service Latency, Serviced Packets Count, Serviced Subscriber Count, Application Liveliness and Session load per Service. KPIs monitoring of these services therefore, play a vital role in optimum usage of network resources such as capacity and throughput. Once we have the attributes like service latency, the network topology can be chosen to provide better quality of experience to the end user. For different services, the attributes may vary and our design takes care of supporting different KPIs for different services model. Additionally, liveliness of application and servers can be monitored using this proposed solution.

Table of Contents

1. Introduction

The TWAMP-Test runs between a Session-Sender and a Session-Reflector RFC 5357 [RFC5357]. The existing TWAMP-Test packet format has existing padding octets that are currently not used (either set to zero or pseudo-random values). These octets can be used to carry additional information between the Session-Sender and the Session-Reflector. The proposed extension uses these padding octets and provide a method to monitor services KPIs in the network. This feature is termed as Services KPI Monitoring using TWAMP.

The services here refers to Layer 4 to Layer 7 services like DPI, SFW, CGNAT, TDF and so on. Additionally these services can refer to applications like DNS, HTTP, FTP and so on. The KPIs MAY include service latency, service load monitoring in terms of number of flows, number of sessions, number of subscribers, number of octets, liveliness of a service.

For instance, Services KPI Monitoring using TWAMP MAY be used to measure service latency of DPI, number of CGNAT flows, number of TDF subscribers and so on. Similarly this MAY be used to monitor the liveliness of the DNS Server, HTTP Server and so on.

As per the proposed extension, both the TWAMP-Control and the TWAMP-Test packet formats are modified. One TWAMP-Test session SHALL be used to monitor KPIs for a specific service. But there can be multiple KPIs monitored using a single test session for a specific service. A single TWAMP-Control connection MAY establish multiple TWAMP-Test sessions that measure KPIs for multiple services in the network.

This extension can be used to monitor KPIs for standalone service or a set of services.

1.1. Conventions used in this document

1.1.1. 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].

1.2. Terminology

TWAMP: Two-Way Active Measurement Protocol

OWAMP: One-Way Active Measurement Protocol

KPI: Key Performance Indicator

DPI: Deep Packet Inspection

CGNAT: Carrier Grade Network Address Translation

SFW: Stateful Firewall

TDF: Traffic Detection Function

DNS: Domain Name Server

HTTP: Hyper Text Transfer Protocol

FTP: File Transfer Protocol

SKMC: Services KPI Monitoring Command

SID: Session ID

PDU: Protocol Data Unit

MBZ: Must Be Zero

HMAC: Hash Message Authentication Code

IPVN: IP Version Number

2. Purpose and Scope

The purpose of this extension is to provide a method to describe an additional optional feature for TWAMP RFC 5357 [RFC5357].

The scope of the extension is limited to specifications of the following features:

1. Extension of the modes of operation through assignment of a new value in the Modes field to communicate feature capability.

2. Addition of new command types to identify, negotiate and monitor the supported services and supported KPIs for each service between Control-Client and TWAMP Server.

3. Use of existing padding octets of TWAMP-Test session to carry the services KPI data that is being monitored as part of the TWAMP-Test session.

The purpose for this feature is to enhance TWAMP protocol to monitor and calculate service-related KPIs in real-time that helps in network performance analysis and optimization.

The actual method to calculate the KPIs is not discussed and depends on implementation.

3. Logical Model

The set of messages that are exchanged between the Control-Client and TWAMP Server to negotiate and monitor the services KPI is referred to as Service Block (Fig 1.) Service Block MAY be a part of the same network element or can be a different entity.

Services KPI-Monitor-REQ is sent from Control-Client to TWAMP Server to get the list of supported services and the KPIs that can be monitored for each service. Once TWAMP Server receives this request, Services KPI-Monitor-RSP is sent with the number of services that can be monitored on this Control-Client connection.

This message is followed by Services KPI-Monitor-IND message from the Session-Reflector which contain a service ID to identify the service and the list of KPIs that are supported for this service.The client replies with the Services KPI-Monitor-ACK message that include the list of KPIs the client is interested in monitoring.These two sets of messages will be repeated for each of the services that Server can monitor.

Then the client will initiate Request-TW-Session Message that contain the service ID for a specific service. Once Server replies with the Accept-Session Message, the client SHALL start sending test packets that MAY contain Service PDU.

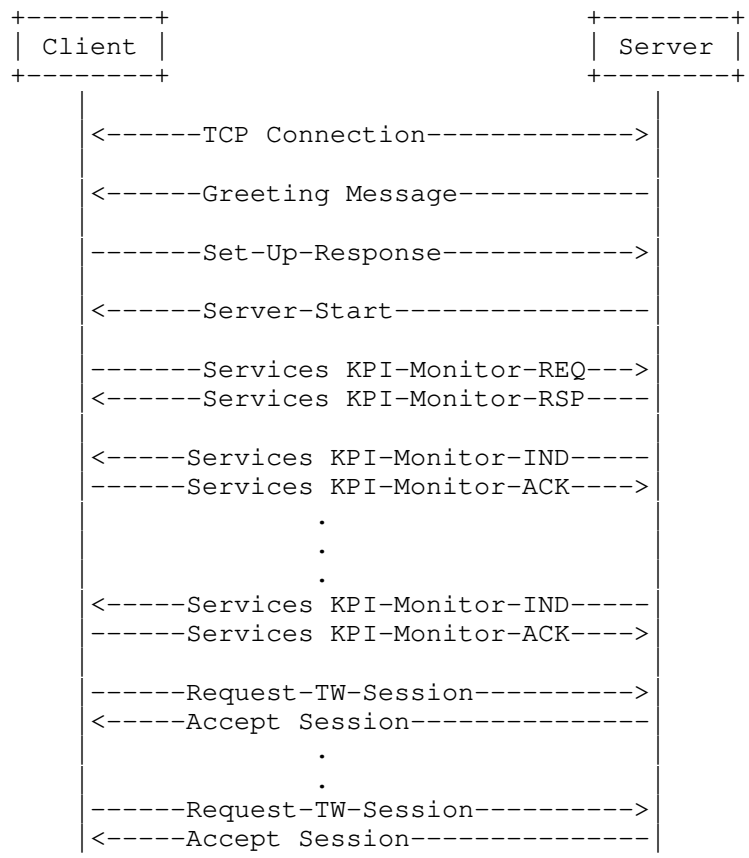


Figure 1

4. TWAMP Extensions

The TWAMP connection establishment follows the procedure defined in Section 3.1 of OWAMP [RFC4656] and Section 3.1 of TWAMP [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 of TWAMP Reflect Octets and Symmetrical Size Features [RFC6038]. The new feature requires a new bit position to identify the ability of a Session-Reflector to monitor Services KPIs. There are changes in both the Control-Client and TWAMP-Test packet formats to support this functionality.

4.1. TWAMP-Control Extensions

The TWAMP-Control is a derivative of the OWAMP-Control, and provides two-way measurement capability. TWAMP; [RFC5357] uses the Modes field to identify and select specific communication capabilities, and this field is a recognized extension mechanism. The following Sections describe one such extension.

4.1.1. Connection Setup with Services KPIs Monitoring

TWAMP-Control connection establishment follows the procedure defined in Section 3.1 of OWAMP; [RFC4656]. The Services KPIs Monitoring using TWAMP mode requires one new bit position (and value) to identify the ability of the Server or the Session-Reflector to monitor the Services KPIs of the sessions. This new extension requires an additional TWAMP mode bit assignment as explained in Section 5.1.

A Control-Client MAY request for Services KPIs monitoring for some of its sessions. To do so, it needs to know which services can be monitored and the corresponding KPIs (of those services)that can be monitored.

Services KPI Monitoring Command (SKMC) consist of a set of messages which SHALL be used for monitoring the KPIs of a service. This new command requires an additional TWAMP Command Number as explained in Section 6.

4.1.2. Services KPI-Monitor-REQ Command

A Control-Client MAY send Services KPI-Monitor-REQ command to the Server to obtain the list of services and their KPIs that can be monitored by the Session-Reflector.

The format of the Services KPI-Monitor-REQ Command is as follows:

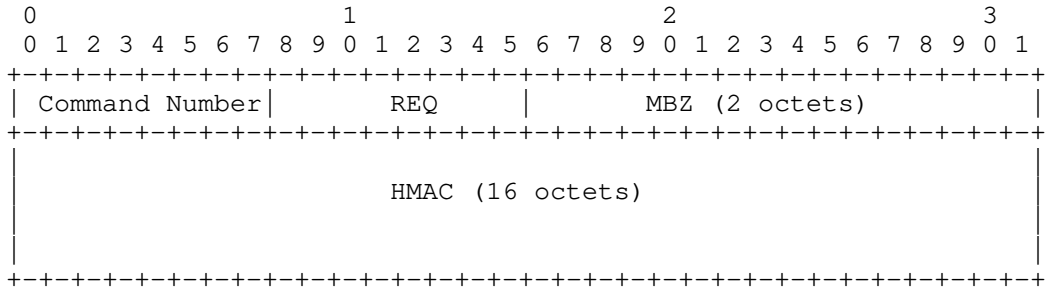


Figure 2: Services KPI-Monitor-REQ Command

Since this is a new command, a Command Number value should be allocated by the IANA in the registry as mentioned in Section 6. The command number indicates that this is one of the Services KPI Monitoring Command. The Control-Client MUST compose this command, and the Server MUST interpret this command, according to the field descriptions below.

The sub-type field MUST be REQ for this message. This message indicates that the Client is requesting Server to send the list of Services and the corrosponding KPIs that can be monitored.

The message is terminated with a single block HMAC, as illustrated above.

The Server MUST respond with Services KPI-Monitor-RSP Command Section 4.1.3.

4.1.3. Services KPI-Monitor-RSP Command

The Server responds to the Services KPI-Monitor-REQ Command by sending a Services KPI-Monitor-RSP Command. The format of the Services KPI-Monitor-RSP Command is as follows:

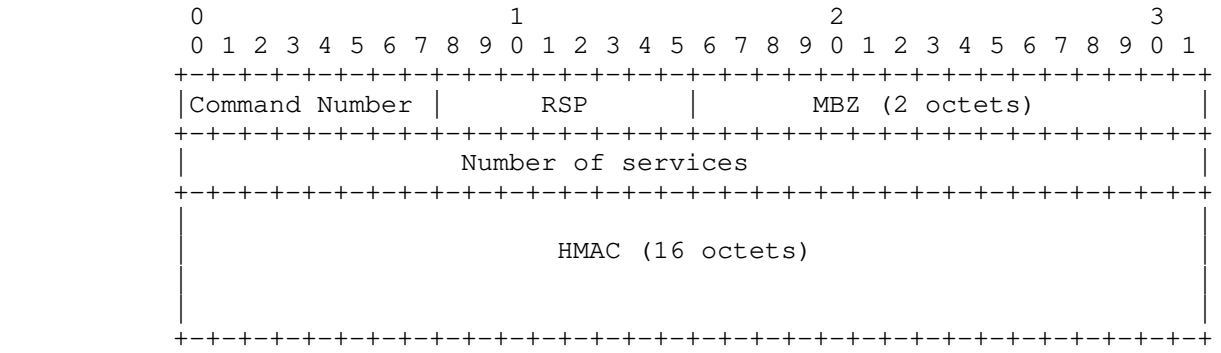


Figure 3: Services KPI-Monitor-RSP Command

The Command Number value here is same as mentioned in Section 6. The Server MUST compose this command, and the Control-Client MUST interpret this command, according to the field descriptions below.

The sub-type field MUST be RSP for this command. The field "Number of Services" indicates the number of Services for which the Session-Reflector can monitor the KPI.

The message is terminated with a single block HMAC, as illustrated above.

4.1.4. Services KPI-Monitor-IND Command

The Server MUST send the Services KPI-Monitor-IND Command after sending Services KPI Monitor-RSP message. This message includes the list of KPIs that can be monitored for a service.

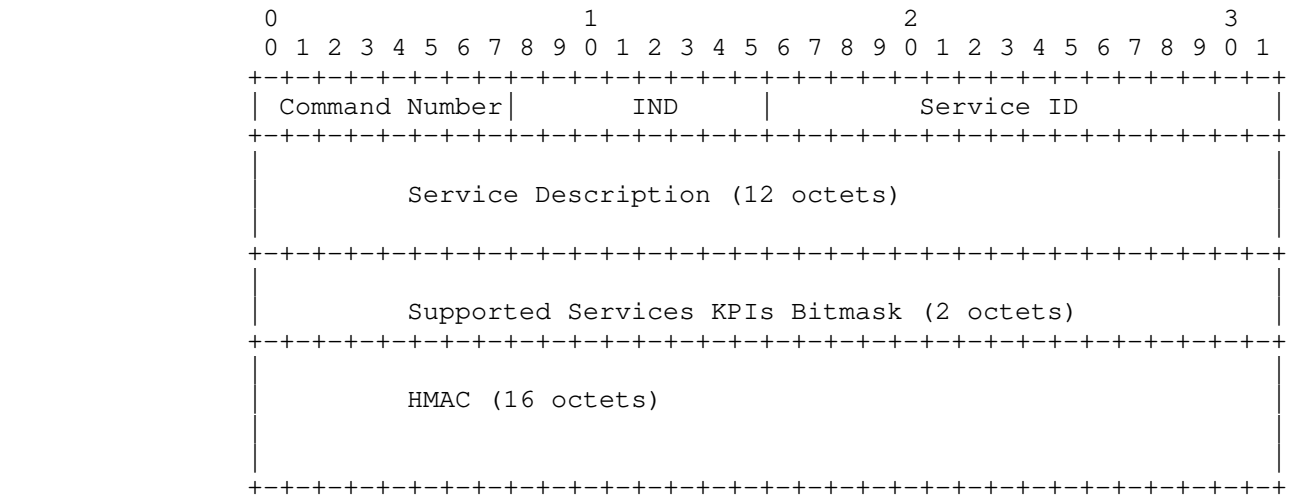


Figure 4: Services KPI-Monitor-IND Command

The Command Number value here is same as mentioned in Section 6. The Server MUST compose this command, and the Control-Client MUST interpret this command, according to the field descriptions below.

The sub-type field MUST be IND for this Command. This field indicates that the Server is responding to the Control-Client with the details of the KPIs of a service that can be monitored by Session-Reflector.

Service ID is an identifier which would be set by the Server to identify a Service. This ID MUST be used in the TWAMP-Control and TWAMP-Test messages to identify a particular Service. The range of Service ID MUST be 1 to 65535; The value 0 is Reserved.

Service Description MAY be set of alphanumeric characters that would provide a brief description of a particular Service. Example: "DPI" "CGNAT" "DNS-Server" "HTTP-Server".

Supported Services KPIs Bitmask is a bitmask that would indicate the kind of KPI Monitoring using TWAMP is supported by the Session-Reflector for a particular Service.

The message is terminated with a single block HMAC, as illustrated above.

For each Services KPIs monitoring supported, the Server MUST send one Services KPI-Monitor-IND Command to the Control-Client.

4.1.5. Services KPI-Monitor-ACK Command

The Control-client MUST respond back with a Services KPI-Monitor-ACK Command for each Services KPI-Monitor-IND Command.

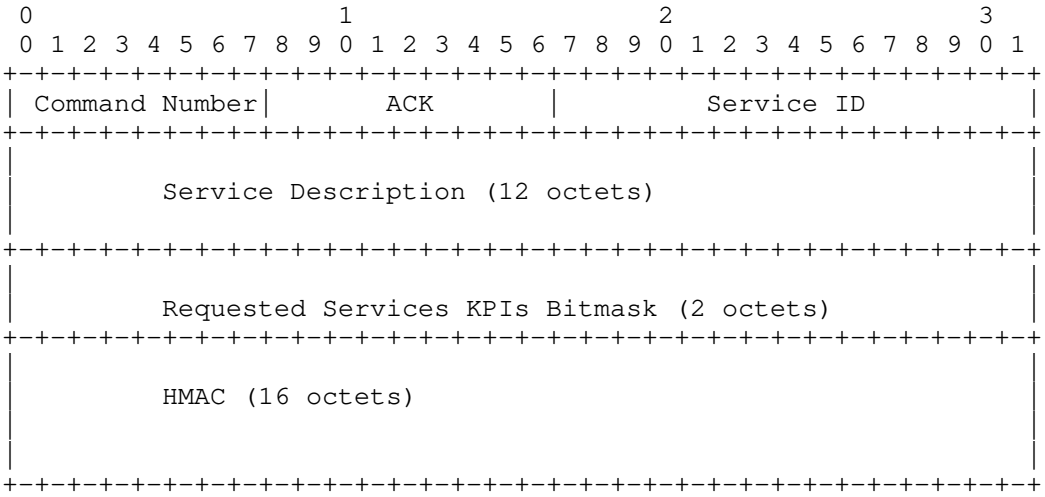


Figure 5: Services KPI-Monitor-ACK Command

The Command Number is same as mentioned in Section 6. The Server MUST frame this command, and the Control-Client MUST interpret this command, according to the field descriptions below.

The sub-type field MUST be ACK for this message. This field indicates that the Control-client is acknowledging the Server with details of which KPIs of a particular service it is interested in.

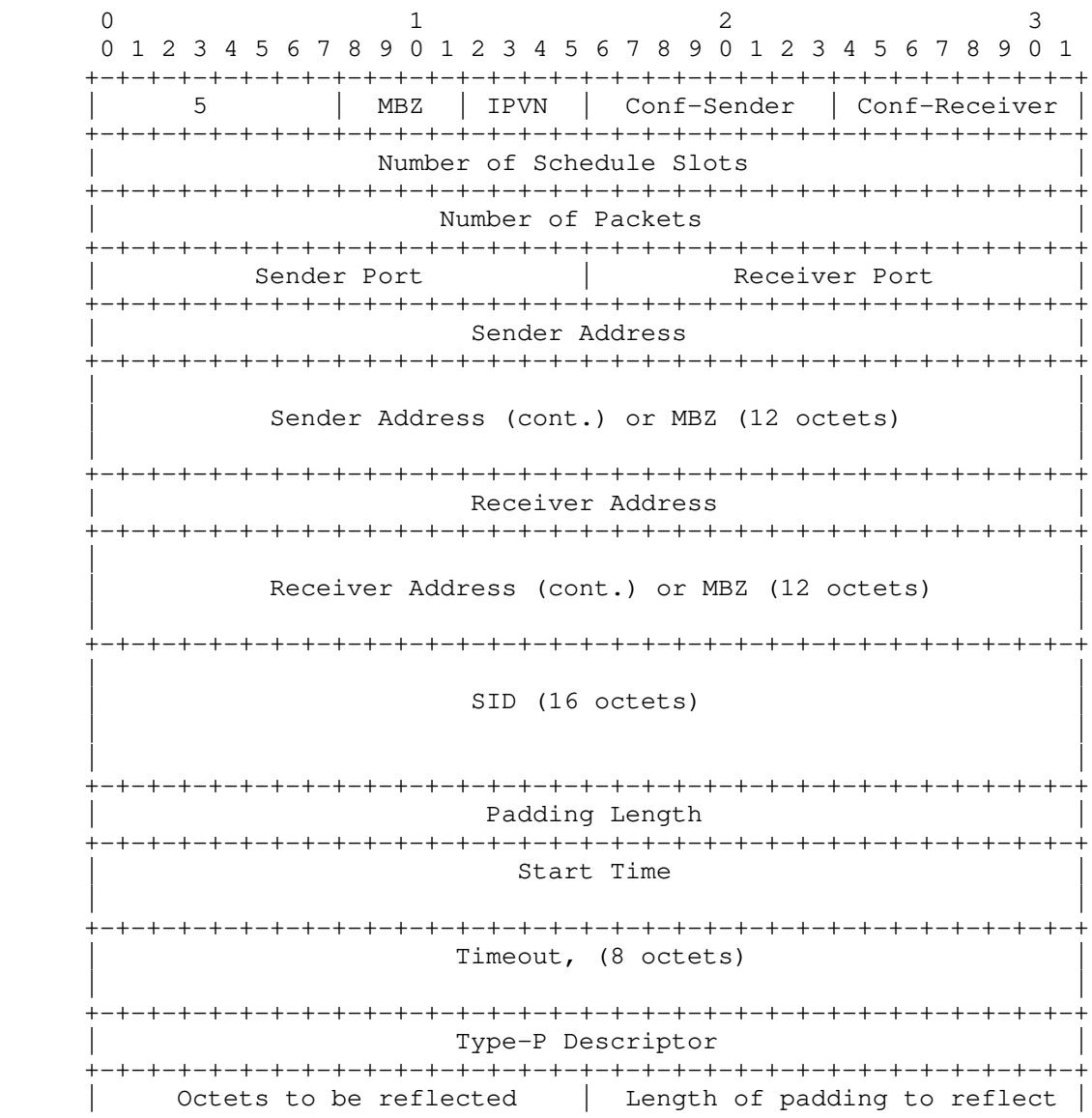
Service ID and Service Description MUST be same as that received in the Services KPI-Monitor-IND Command. These two fields together identify a particular Service.

Requested Services KPIs Bitmask MUST be set by the Control-Client and that indicates the KPIs of the services that the Control-Client is interested in monitoring. The KPIs can be a subset or the full set of KPIs sent in the corresponding Service KPI-Monitor-IND Command.

The Command is terminated with a single block HMAC, as illustrated above.

For each Services KPI-Monitor-IND Command received at the control-client, it acknowledges by sending a Services KPI-Monitor-ACK Command.

4.1.6. Request-TW-Session Command Format



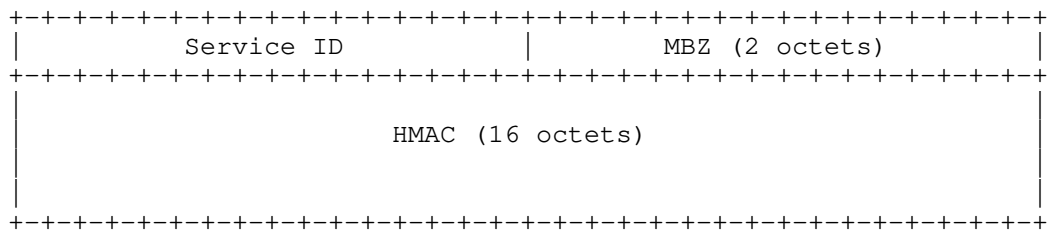


Figure 6: Request-TW-Session Command

This new feature requires 2 octets to indicate the Service ID as a part of Request-TW-Session Command. See Section 6 for details on the octet position. If Services KPIs Monitoring using TWAMP feature is not requested as a part of this TWAMP-Test Session, then the Service ID MUST be 0.

If Service ID has a non-zero value, then the Padding Length field MAY not have any significance. The test packets between Session-Sender and Session-Reflector MAY be of different size based on the implementation.

The actual test packets MAY contain valid data which SHOULD be interpreted by Session-Sender or Session-Reflector to monitor Services KPIs. Please refer TWAMP-Test Extension Section 4.2 for more details.

4.2. TWAMP-Test Extension

As part of this extension, the existing Packet Padding octets in the Test Packet MAY be used for the monitoring of the Services KPIs as explained in KPI Implemnetation Draft. The Packet Padding octets which were either zero or filled with pseudo-random values MAY now have some valid data like timestamps, statistics, service PDUs and so on.

The Session-Sender Test Session data packet formats are provided below.

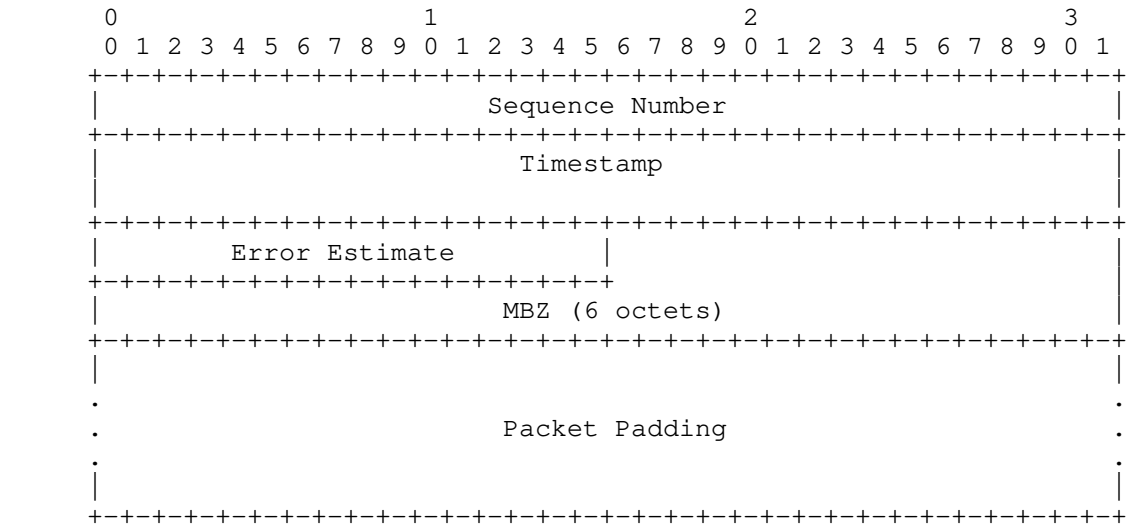


Figure 7: Unauthenticated Mode Session-Sender Data Packet Format

As a part of the extension, 6 octets of MBZ are added after the Error Estimate field.

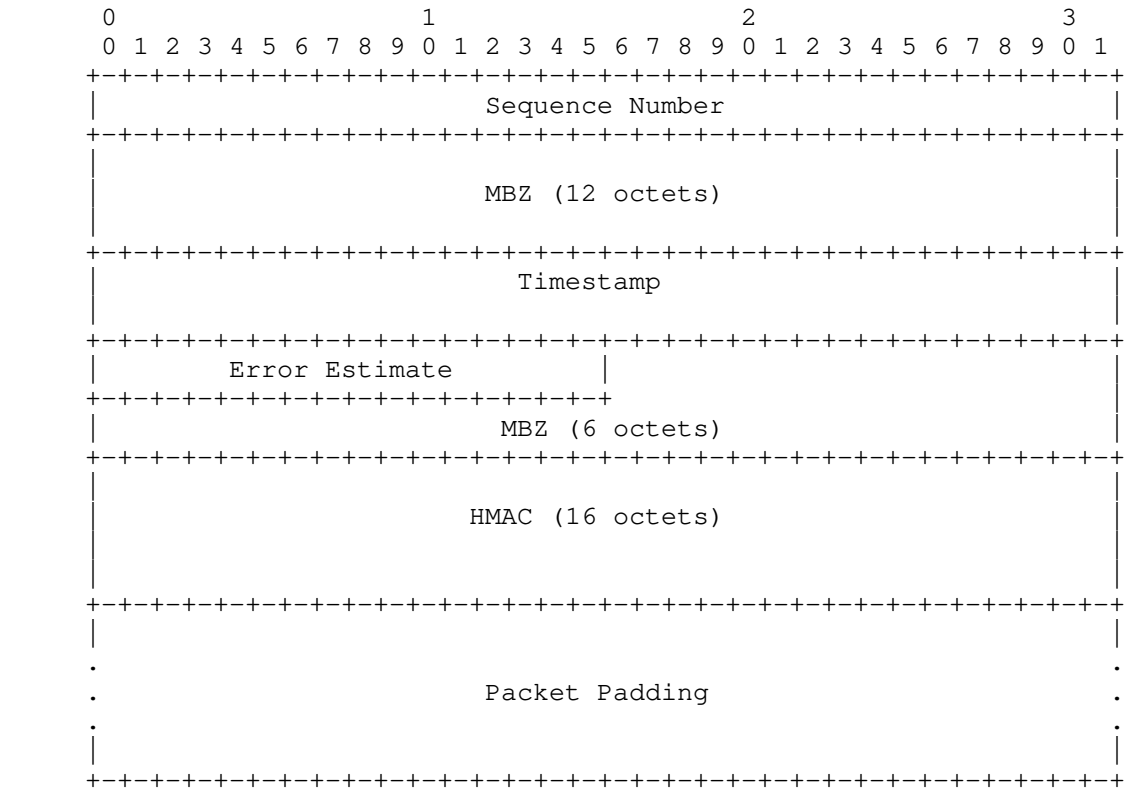


Figure 8: Authenticated and Encrypted Mode Session-Reflector Data Packet Format

The Session-Reflector Test Session data packet formats are provided below.

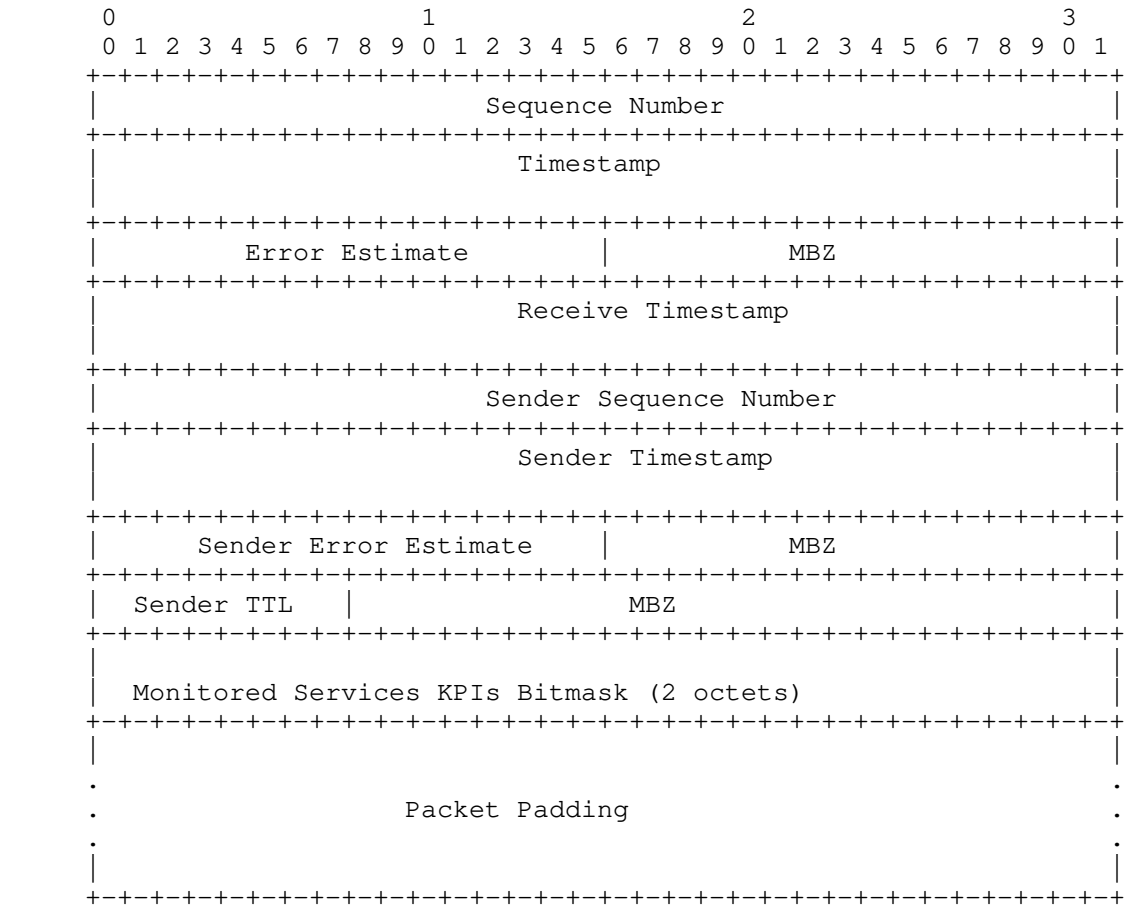
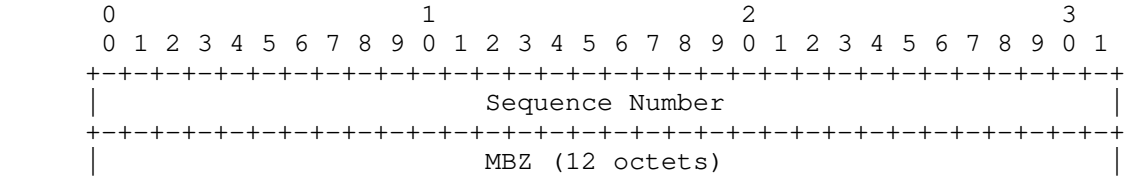


Figure 9: Unauthenticated Mode Session-Reflector Data Packet Format

As a part of this extension, The 3 octets of MBZ are added after the Error Estimate field to align the next set of fields.

Monitored Services KPIs Bitmask indicates the services KPIs that are present in this message. The KPIs would be present in the Packet Padding area in the same order as indicated by Bitmask starting from bit 0 Position.



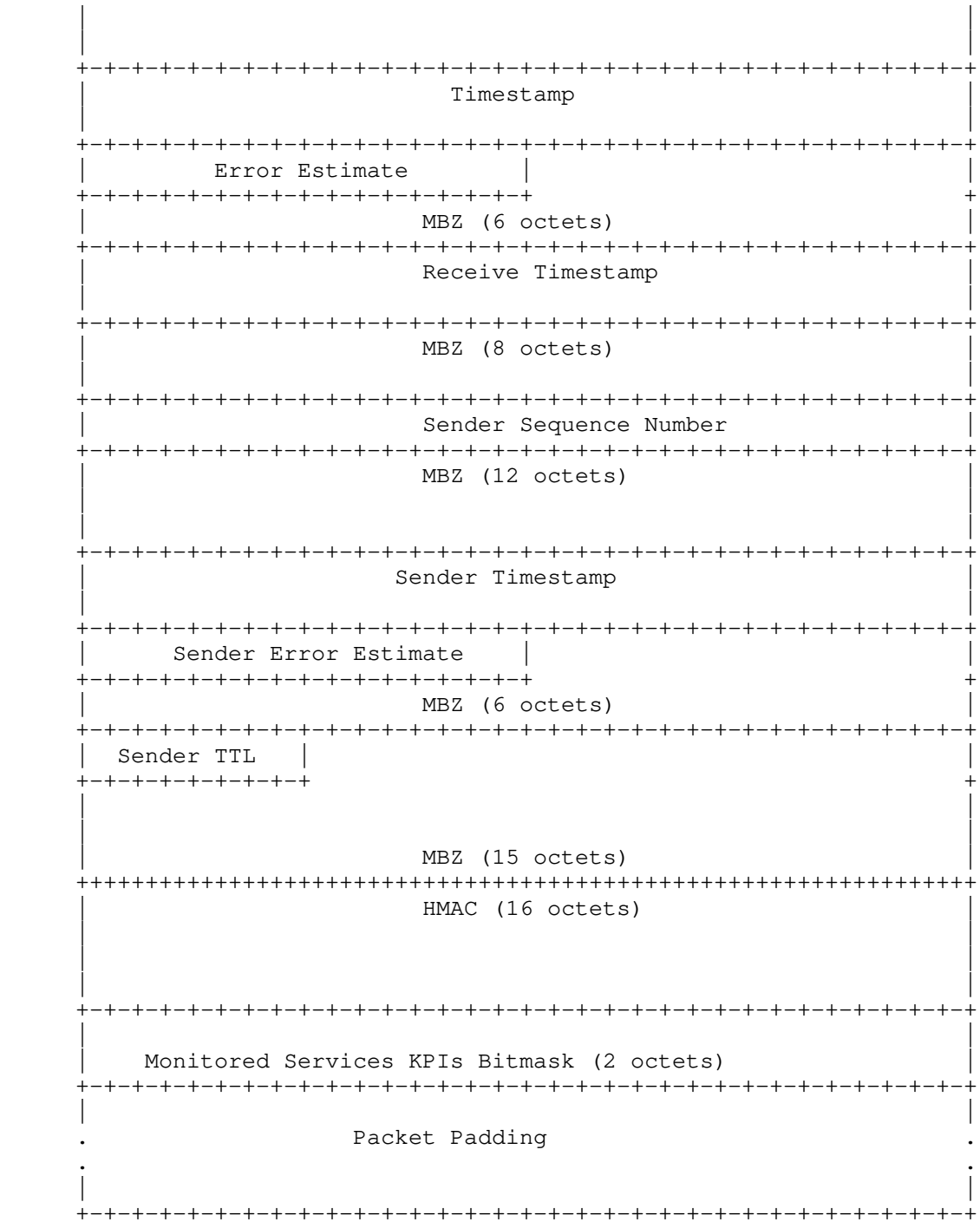

Timestamp
Error Estimate
MBZ (6 octets)
Receive Timestamp
MBZ (8 octets)
Sender Sequence Number
MBZ (12 octets)
Sender Timestamp
Sender Error Estimate
MBZ (6 octets)
Sender TTL
MBZ (15 octets)
HMAC (16 octets)
Monitored Services KPIs Bitmask (2 octets)
Packet Padding

Figure 10: Authenticated and Encrypted Mode Session-Reflector Data Packet Format

As a part of this extension, Monitored Services KPIs Bitmask indicates the services KPIs that are present in this message. The KPIs would be present in the Packet Padding area in the same order as indicated by Bitmask starting from bit 0 Position. The set of KPIs defined for a service are listed in KPI Implementation draft

5. Acknowledgements

We would like to thank Perceival A Monteiro for their comments, suggestions, reviews, helpful discussion, and proof-reading

6. IANA Considerations

The TWAMP-Modes registry defined in [RFC6038]. IANA is requested to reserve a new bit in Modes registry for Services KPIs Monitoring Capability as follows:

Value	Description	Semantics	Reference
X (proposed 256)	Services KPIs Monitoring Capability	bit position Y(proposed 8)	This document

Table 1: Services KPIs Monitoring Capability

TWAMP-Control Command Number Registry defined in [RFC5938].IANA is requested to reserve a Command Number for Services KPIs Monitoring Capability as follows:

Value	Description	Semantics	Reference
SKMC (proposed 11)	Services KPIs Monitoring Command		This document

Table 2: New Service Latency Monitoring Command

TWAMP Services KPIs sub-type Registry

IANA is requested to reserve and maintain the sub-types as a part of Services KPIs Monitoring Command as follows:

Value	Description	Explanation
0	Reserved	
1	REQ	Section 4.1.2
2	RESP	Section 4.1.3
3	IND	Section 4.1.4
4	ACK	Section 4.1.5

Table 3: TWAMP Services KPIs sub-type Registry

TWAMP Services KPIs Registry

IANA is requested to reserve and maintain the below Services KPIs:

Value	Description	Explanation
0	None	
1	Keepalive	Whether the respective service is running or not
2	Service Latency	Service Latency which SHALL include the transit time and actual service time
4	Serviced Packets Count	Number of ingress and egress packets for the respective service
8	Serviced Bytes Count	Number of ingress and egress bytes for the respective service.
16	Serviced Subscriber Count	Number of subscribers currently active for the respective service.

Table 4: TWAMP Services KPIs Registry

Request-TW-Session message defined in [RFC6038].IANA is requested to reserve 2 octets for Service ID as follows:

Value	Description	Semantics	Reference
X	Service ID	2 Octets starting from offset 92th Octet	This document

Table 5: New Services KPIs Monitoring Capability

7. Security Considerations

NA

8. 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>.

[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>.

[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>.

[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

Srivathsa Sarangapani
Juniper Networks
Bangalore 560079
INDIA

Phone: +91 9845052354
Email: srivathsas@juniper.net

Peyush Gupta
Juniper Networks
T-313, Keerti Royal Apartment, Outer Ring Road
Bangalore, Karnataka 560043
INDIA

Phone: +91 9449251927
Email: peyushg@juniper.net

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

Phone: +91 944984401
Email: vinayakh@gmail.com