

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

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

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

1. Introduction

The Two-Way Active Measurement Protocol (TWAMP) [RFC5357] is used to measure network performance parameters such as latency, bandwidth, and packet loss by sending probe packets and measuring their experience in the network. To date, TWAMP implementations do not come with a standard management framework and, as such, configuration depends on the various proprietary mechanisms developed by the corresponding TWAMP vendor. This document addresses this gap by formally specifying the TWAMP data model using YANG.

1.1. Motivation

In current TWAMP deployments, the lack of a standardized data model limits the flexibility to dynamically instantiate TWAMP-based measurements across equipment from different vendors. In large, virtualized, and dynamically instantiated infrastructures where network functions are placed according to orchestration algorithms as discussed in [I-D.unify-nfvrg-challenges][I-D.unify-nfvrg-devops], proprietary mechanisms for managing TWAMP measurements pose severe limitations with respect to programmability.

Two major trends call for revisiting the standardization on TWAMP management aspects. First, we expect that in the coming years large-scale and multi-vendor TWAMP deployments will become the norm. From an operations perspective, dealing with several vendor-specific TWAMP configuration mechanisms is simply unsustainable in this context. Second, the increasingly software-defined and virtualized nature of network infrastructures, based on dynamic service chains [NSC] and programmable control and management planes [RFC7426] requires a well-defined data model for TWAMP implementations. This document defines such a TWAMP data model and specifies it formally using the YANG data modeling language [RFC6020].

1.2. Terminology

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

1.3. Document Organization

The rest of this document is organized as follows. Section 2 presents the scope and applicability of this document. Section 3 provides a high-level overview of the TWAMP data model. Section 4 details the configuration parameters of the data model and Section 5 specifies in YANG the TWAMP data model. Section 6 lists illustrative

examples which conform to the YANG data model specified in this document. Appendix A elaborates these examples further.

2. Scope, Model, and Applicability

The purpose of this document is the specification of a vendor-independent data model for TWAMP implementations.

Figure 1 illustrates a redrawn version of the TWAMP logical model found in Section 1.2 of [RFC5357]. The figure is annotated with pointers to the UML diagrams provided in this document and associated with the data model of the four logical entities in a TWAMP deployment, namely the TWAMP Control-Client, Server, Session-Sender and Session-Reflector. As per [RFC5357], unlabeled links in Figure 1 are unspecified and may be proprietary protocols.

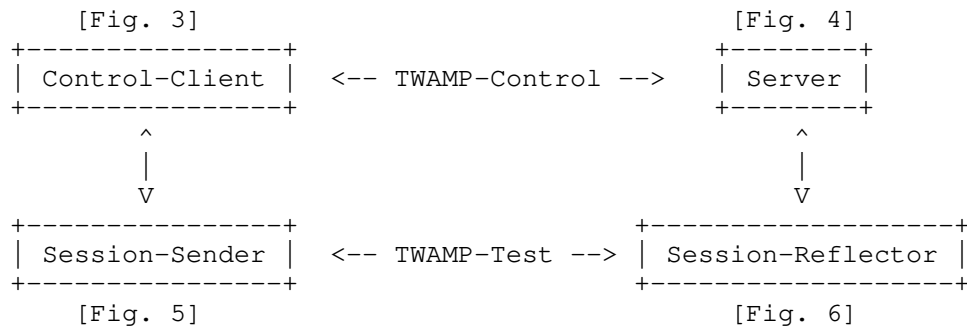


Figure 1: Annotated TWAMP logical model

As per [RFC5357], a TWAMP implementation may follow a simplified logical model, in which the same node acts both as the Control-Client and Session-Sender, while another node acts at the same time as the TWAMP Server and Session-Reflector. Figure 2 illustrates this simplified logical model and indicates the interaction between the TWAMP configuration client and server using, for instance, NETCONF [RFC6241] or RESTCONF [I-D.ietf-netconf-restconf]. Note, however, that the specific protocol used to communicate the TWAMP configuration parameters specified herein is outside the scope of this document. Appendix B considers TWAMP operational commands, which are also outside the scope of this document.

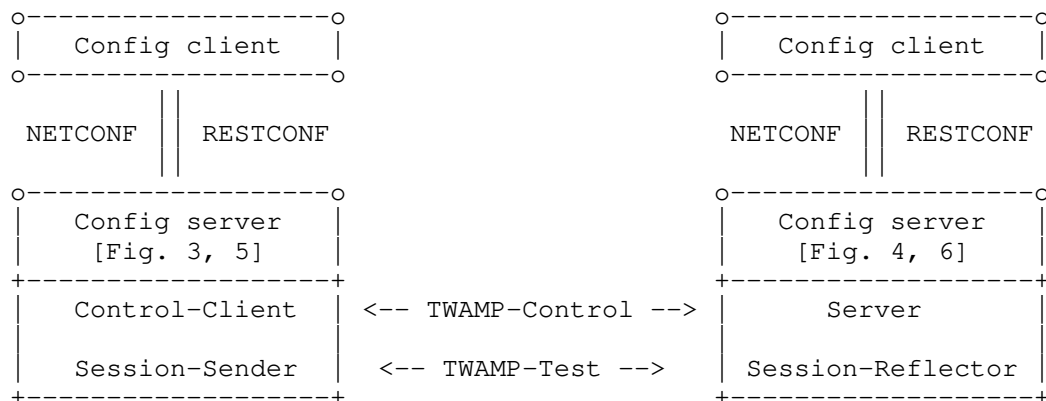


Figure 2: Simplified TWAMP model and protocols

3. Data Model Overview

A TWAMP data model includes four categories of configuration items. Global configuration items relate to parameters that are set on a per device level. For example, the administrative status of the device with respect to whether it allows TWAMP sessions and, if so, in what capacity (e.g. Control-Client, Server or both), are typical instances of global configuration items. A second category includes attributes that can be configured on a per control connection basis, such as the Server IP address. A third category includes attributes related to per test session attributes, for instance setting different values in the Differentiated Services Code Point (DSCP) field. Finally, the data model could include attributes that relate to the operational state of the TWAMP implementation.

As we describe the TWAMP data model in the remaining sections of this document, readers should keep in mind the functional entity grouping illustrated in Figure 1.

3.1. Control-Client

A TWAMP Control-Client has an administrative status field set at the device level that indicates whether the node is enabled to function as such.

Each TWAMP Control-Client is associated with zero or more TWAMP control connections. The main configuration parameters of each control connection are:

- o A name which can be used to uniquely identify at the Control-Client a particular control connection. This name is necessary

for programmability reasons because at the time of creation of a TWAMP control connection not all IP and TCP port number information needed to uniquely identify the connection is available.

- o The IP address of the interface the Control-Client will use for connections
- o The IP address of the remote Server
- o 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:

- o 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.
- o The IP address and UDP port number of the Session-Sender of the path under test by TWAMP
- o The IP address and UDP port number of the Session-Reflector of said path
- o 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:

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

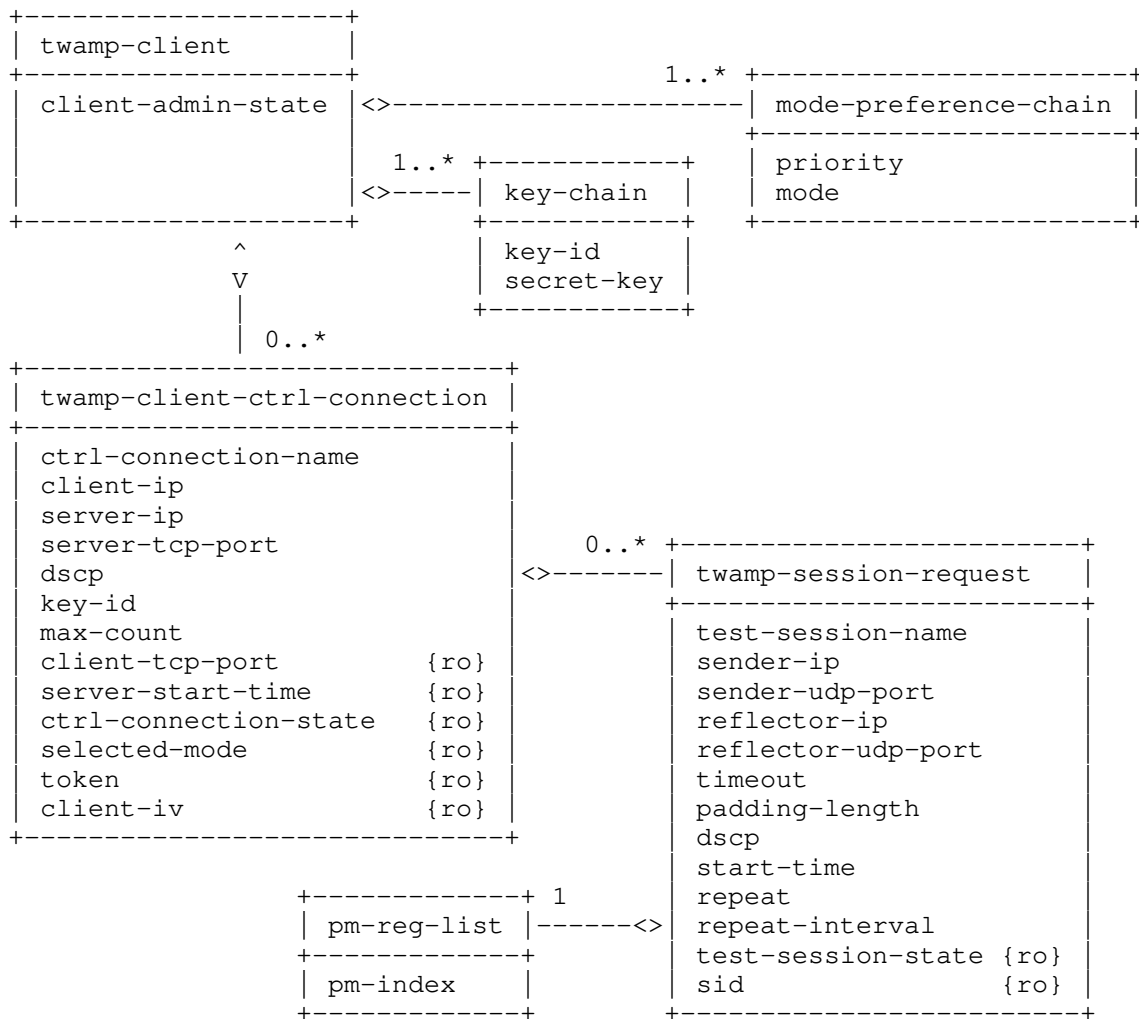


Figure 3: TWAMP Control-Client UML class diagram

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 advertised 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).

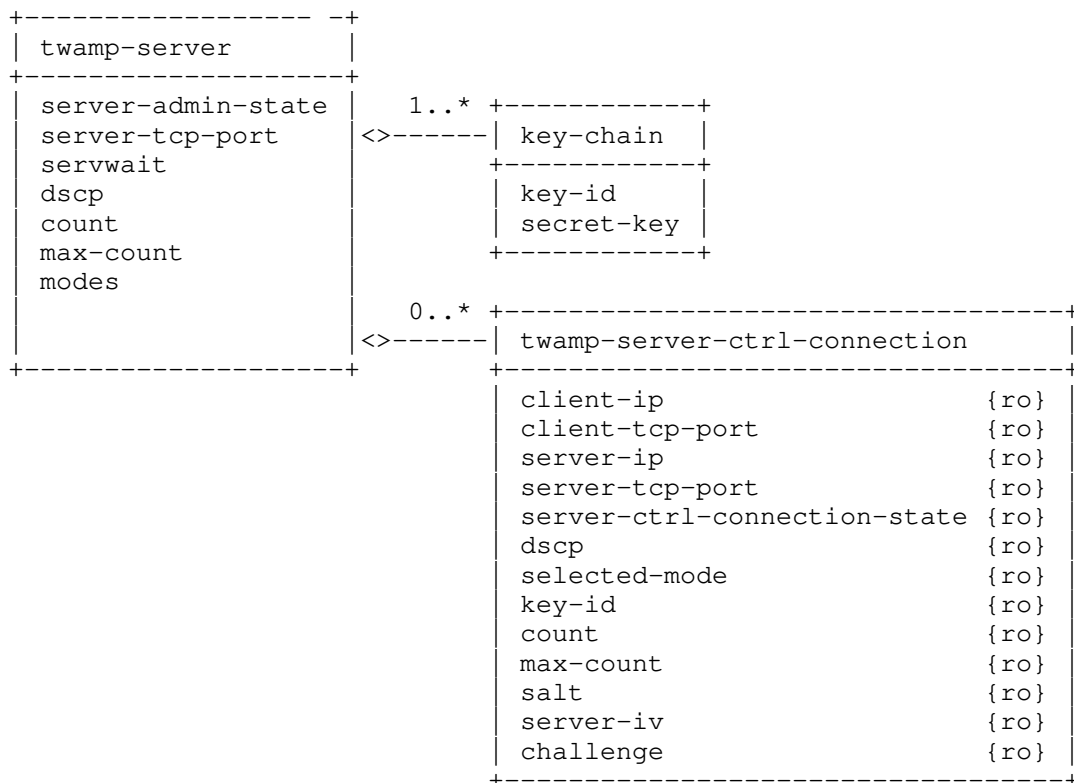


Figure 4: TWAMP Server UML class diagram

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.

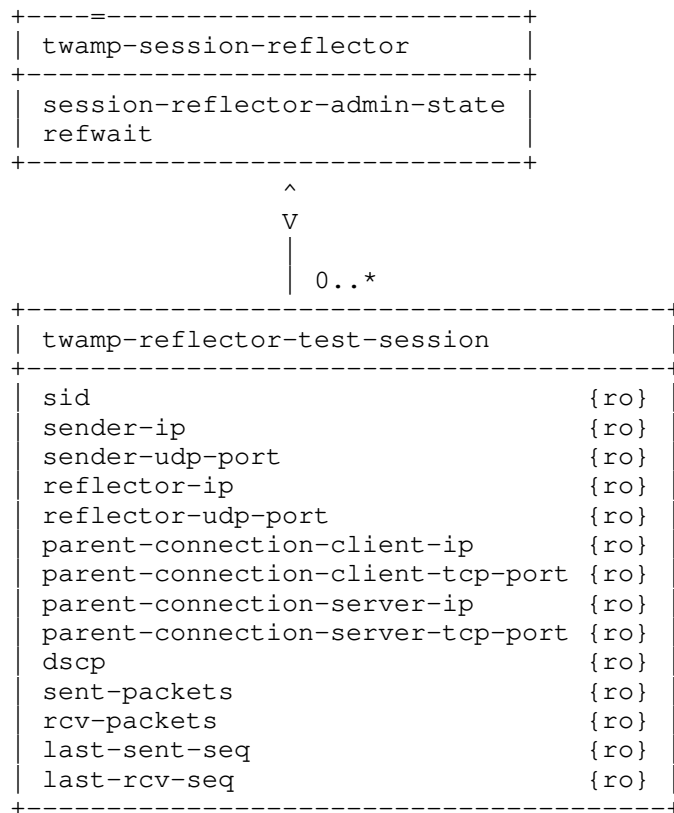


Figure 6: TWAMP Session-Reflector UML class diagram

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>
      </twamp-session-request>
    </twamp-client-ctrl-connection>
  </twamp-client>
</twamp>
```

```
        <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

Network Working Group
Internet-Draft
Updates: 2330 (if approved)
Intended status: Informational
Expires: April 16, 2016

J. Fabini
TU Wien
A. Morton
AT&T Labs
October 14, 2015

Updates for IPPM's Framework: Timestamping and Use Cases
draft-fabini-ippm-2330-time-00

Abstract

Quality and accuracy of measurements depend on the selection of appropriate locations and timers for timestamp acquisition. This memo updates the IP Performance Metrics (IPPM) Framework RFC 2330 with new considerations on timers, timestamps and time-related definitions with particular focus on wireless networks and virtualized hosts.

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 April 16, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Scope	3
3. Definition and Use of Wire Time	4
3.1. Virtualization	5
4. Definition and Use of Host Time	5
4.1. Virtualization	6
5. Encryption	6
6. Security Considerations	6
7. IANA Considerations	7
8. Acknowledgements	7
9. References	7
9.1. Normative References	7
9.2. Informative References	9
Authors' Addresses	9

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].

Time is fundamental to measurements. The IPPM framework [RFC2330] includes a comprehensive terminology, definition and discussion of time- and clock-related concepts. But network technology has evolved since 1998. First, host computing performance and software complexity have increased while network delays have decreased. These advances enable development of new concepts like host virtualization in cloud computing, and network function virtualization. One side-effect of these technologies is that packets spend more and more time in software, their propagation through the stack being biased by a series of uncertainty factors like buffers, queues and operating system scheduling. Measurements can acquire timestamps in various locations within the host's software stack, causing potentially

significant variances in timestamp values [MCDC]. This is a consequence of just one alternative that the IPPM framework [RFC2330] defines for these timestamps: host time. This memo seeks to update this aspect of the IPPM framework by discussing alternatives and proposing generic solutions that can be referenced by metric definitions.

Second, the IPPM framework was approved one year before the release of the first IEEE 802.11 series standard. This was a time when the data communication landscape was dominated by wired communication technologies and wireless data communications were in their infancy. This fact originated IPPM definitions like host time or wire time. Today, after more than one decade and half, wireless IEEE 802.11 networks and cellular packet-switched data technologies like High-Speed Packet Access (HSPA) and Long Term Evolution (LTE) are responsible for an essential part of the worldwide Internet access traffic. This raises the question, how the IPPM framework concept of wire time can be mapped to wireless networks.

This memo revisits and updates time-related IPPM framework [RFC2330] definitions like host time and wire time in the light of technological advances.

2. Scope

The purpose of this memo is to update the IPPM metrics framework [RFC2330] with respect to timestamps and virtualization. Concepts like "host time" or "wire time" must be revisited to be applicable for wireless networks and virtualized environments.

The scope is to update key sections of [RFC2330] ...

Potential Topics (Draft outline for discussion):

- Fundamentals

1. Host Time (further develop the definition as mentioned but not defined in RFC2330. Explain variants, propose examples and guidance). Dedicated subsection and challenge: virtualization.

2. Wire Time (rename "Wire Time" to "Medium Time" or "Media Time" to account for wireless networks; revisit the definition of "Media Arrival Time" and "Media Exit Time" in the light of encryption, complex network coding, interleaving etc. Define how to measure media time in wireless networks) Dedicated subsection and challenge: virtualization

3. Define new terms wrt timestamps that help to differentiate between software delays within a host. Ideally it should give guidance and define terms that allow to differentiate between delays (uncertainties) that a) are caused by host load, timer resolution, system scheduling, etc. and b) the ones that originate when packets wait in drivers because of network access policies or network congestion. A "network (or IP) timestamp" can be useful in (virtualized) environments it might make sense to define the term "network timestamp" as "tcpdump timestamp of the host instance that is located closest to the hardware". Hypervisor tcpdump timestamps should be preferred over VM timestamps.

4. (optional) Protocol design considerations: (Security vs. flexibility): adding timestamps into headers/payload protocol fields at driver level (tcpdump equivalent) is difficult/impossible if link is protected using SSL/TLS/IPsec. Meaningful: insert timestamp in application space.

5. (probably not in IPPM) NTP is understood to operate sub-optimally for time-transfer to virtual machines (VM) as guest-clients in some hosts. What mechanisms can be employed to improve time accuracy and stability when VMs intend to perform time measurement?

3. Definition and Use of Wire Time

[RFC2330] defines the following terms related to wire time (defined in terms of an Internet host H observing an Internet link L at a particular location):

- o For a given 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
- o For a given 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.

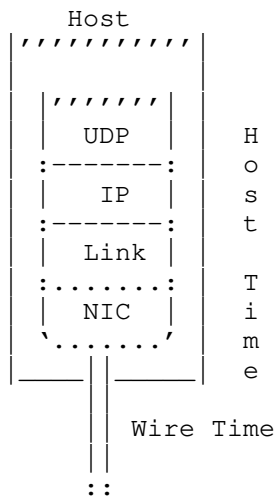
However, wireless LAN and cellular networks are essential components of today's Internet. This memo proposes to replace the term "wire time" by the equivalent term "media time" and provides additional guidance for wireless network measurements.

This definition of media time raises one additional challenge: how can "media arrival time" and "media exit time" be defined for wireless networks. Aligning with existing physical-layer standards of other standardization organizations like the 3GPP, this memo recommends the antenna connector of the Host(?) to be used for measuring media time in wireless networks.

3.1. Virtualization

Virtualization adds additional level of uncertainty for timestamps: VM application, VM tcpdump, Hypervisor tcpdump. What about wire time for virtual network interfaces? Should it be bound to the related physical interface or to the hypervisor/virtual interface?

There was some flexibility in the original mention of Host Time. Virtualization adds more layers where host timestamps could be applied or derived, and the specification requires its own framework:



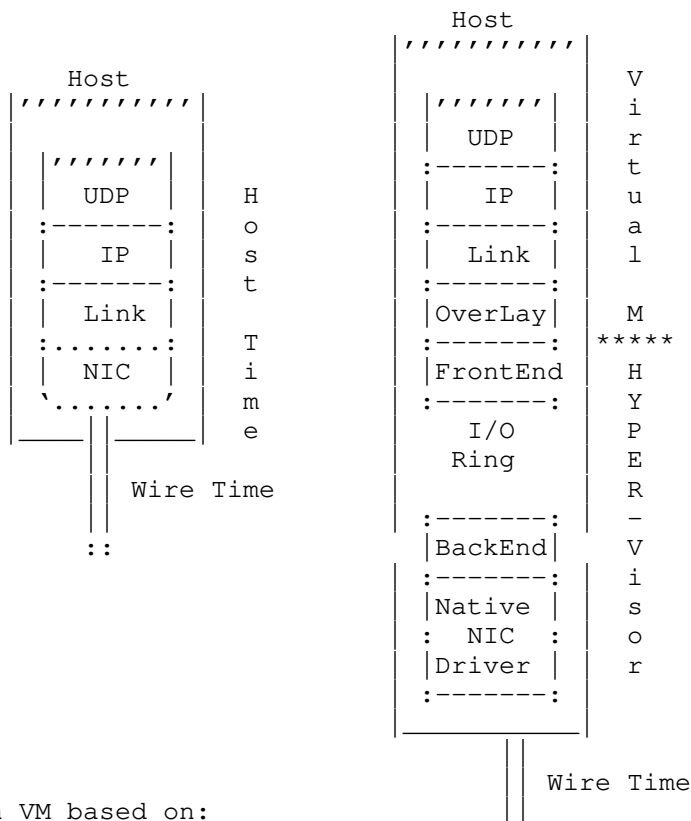
4. Definition and Use of Host Time

Section 3 of [RFC2330] includes an occurrence of the term "host time" but does not define it. Some metric definitions like one-way delay in section 3.7.2 of [RFC2679] or round-trip delay in section 2.7 of [RFC2681] discuss the consequences of differences between wire time and host time. Common to these metric definitions is the request to report errors and uncertainties resulting from the difference between host time and wire time.

However, the evolution of network technology has decreased the network delay substantially. Today, the delay of wireless and wired LAN links are in the same order of magnitude as the uncertainty of host time, the difference between timestamps at application layer and the ones in kernel space being substantial as pointed out in [MCDC]). Instead of accepting this uncertainty as an inherent part of measurements, this memo recommends to accurately define the location within the stack where timestamps are assigned to a packet.

4.1. Virtualization

Virtualization extends the concept of host time beyond earlier limits. For instance packet captures (tcpdump) in virtualized environments are possible at VM-level and at hypervisor level. Therefore, the definition of "Host Time" spans a broad range of timestamps, encompassing anything from timestamps acquired by an application running on top of a VM, down to tcpdump timestamps acquired within the VM or hypervisor network stack.



Xen VM based on:

<http://www.cc.gatech.edu/~lingliu/papers/2012/YiRen-CollaborateCom2012.pdf>

5. Encryption

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 ...

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>>.
- [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>>.

- [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>>.
- [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

- [MCDC] Fabini, J. and T. Zseby, "M2M communication delay challenges: Application and measurement perspectives", IEEE International Instrumentation and Measurement Technology Conference (I2MTC 2015) doi: 10.1109/I2MTC.2015.7151564, May 2015.
- [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

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/>

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/>

INTERNET-DRAFT

N. Elkins
Inside Products
R. Hamilton
Chemical Abstracts Service
M. Ackermann
BCBS Michigan
June 26, 2017

Intended Status: Proposed Standard
Expires: December 28, 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.

Status of this Memo

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

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

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

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

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

Copyright and License Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

IETF Trust Legal Provisions of 28-dec-2009, Section 6.b(i), paragraph 3: This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Background	5
1.1	Terminology	5
1.2	Rationale for defined solution	5
1.3	IPv6 Transition Technologies	6
2	Measurement Information Derived from PDM	6
2.1	Round-Trip Delay	6
2.2	Server Delay	7
3	Performance and Diagnostic Metrics Destination Option Layout	7
3.1	Destination Options Header	7
3.2	Performance and Diagnostic Metrics Destination Option	7
3.2.1	PDM Layout	7
3.2.2	Base Unit for Time Measurement	10
3.3	Header Placement	11
3.4	Header Placement Using IPsec ESP Mode	11
3.4.1	Using ESP Transport Mode	11
3.4.2	Using ESP Tunnel Mode	12
3.5	Implementation Considerations	12
3.5.1	PDM Activation	12
3.5.2	PDM Timestamps	12
3.6	Dynamic Configuration Options	12
3.7	Information Access and Storage	13
4	Security Considerations	13
4.1	Resource Consumption and Resource Consumption Attacks	13
4.2	Pervasive monitoring	13
4.3	PDM as a Covert Channel	14
4.4	Timing Attacks	14
5	IANA Considerations	15
6	References	15
6.1	Normative References	15
6.2	Informative References	16
	Appendix A: Context for PDM	16
A.1	End User Quality of Service (QoS)	16
A.2	Need for a Packet Sequence Number (PSN)	17
A.3	Rationale for Defined Solution	17
A.4	Use PDM with Other Headers	17
	Appendix B : Timing Considerations	19
B.1	Timing Differential Calculations	19
B.2	Considerations of this time-differential representation	20
B.2.1	Limitations with this encoding method	20
B.2.2	Loss of precision induced by timer value truncation	21
	Appendix C: Sample Packet Flows	22
C.1	PDM Flow - Simple Client Server	22
C.1.1	Step 1	23
C.1.2	Step 2	23
C.1.3	Step 3	24
C.1.4	Step 4	25

C.1.5 Step 5	26
C.2 Other Flows	26
C.2.1 PDM Flow - One Way Traffic	26
C.2.2 PDM Flow - Multiple Send Traffic	28
C.2.3 PDM Flow - Multiple Send with Errors	29
Appendix D: Potential Overhead Considerations	30
Acknowledgments	31
Authors' Addresses	32

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

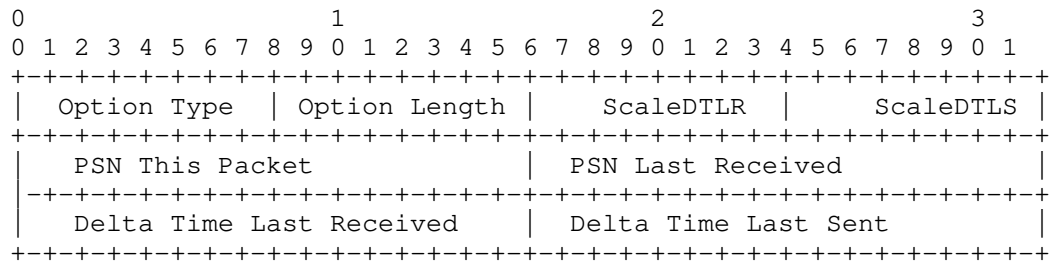
The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) contains the following fields:

SCALEDTLR: Scale for Delta Time Last Received
SCAEDTLS: 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 SCAEDTLS.

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 may want to use PDM with other headers (L2, L3, etc). For example, if PDM is used 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 your 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 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

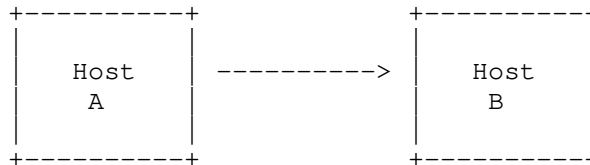
Following is a sample simple flow for the PDM with one packet sent from Host A and one packet received by Host B. The PDM does not require time synchronization between Host A and Host B. The calculations to derive meaningful metrics for network diagnostics are shown below each packet sent or received.

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:

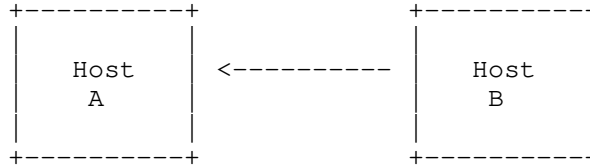
Packet Sequence Number of the last packet received:	25
Time the last packet was received:	11:00:03
Packet Sequence Number of this packet:	12
Time this packet is being sent:	11:00:07

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:

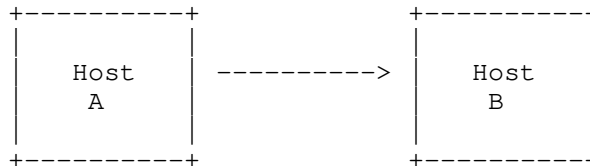
Round trip delay = 12 - 4 or 8

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	PSN	Delta Time	Delta Time
		This Packet	Last Recvd	Last Recvd	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: Informational
Expires: July 24, 2016

A. Morton
AT&T Labs
January 21, 2016

Active and Passive Metrics and Methods (and everything in-between, or
Hybrid)
draft-ietf-ippm-active-passive-06

Abstract

This memo provides clear definitions for Active and Passive performance assessment. The construction of Metrics and Methods can be described as Active or Passive. Some methods may use a subset of both active and passive attributes, and we refer to these as Hybrid Methods. This memo also describes multiple dimensions to help evaluate new methods as they emerge.

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 July 24, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
2. Purpose and Scope	3
3. Terms and Definitions	3
3.1. Performance Metric	4
3.2. Method of Measurement	4
3.3. Observation Point	4
3.4. Active Methods	4
3.5. Active Metric	5
3.6. Passive Methods	5
3.7. Passive Metric	6
3.8. Hybrid Methods and Metrics	6
4. Discussion	8
4.1. Graphical Representation	8
4.2. Discussion of PDM	10
4.3. Discussion of "Coloring" Method	11
4.4. Brief Discussion of OAM Methods	11
5. Security considerations	12
6. IANA Considerations	12
7. Acknowledgements	12
8. References	13
8.1. Normative References	13
8.2. Informative References	14
Author's Address	15

1. Introduction

The adjectives "active" and "passive" have been used for many years to distinguish two different classes of Internet performance assessment. The first Passive and Active Measurement (PAM) Conference was held in 2000, but the earliest proceedings available on-line are from the second PAM conference in 2001 [<https://www.ripe.net/ripe/meetings/pam-2001>].

The notions of "active" and "passive" are well-established. In general:

An Active metric or method depends on a dedicated measurement packet stream and observations of the stream.

A Passive metric or method depends **solely** on observation of one or more existing packet streams. The streams only serve

measurement when they are observed for that purpose, and are present whether measurements take place or not.

As new techniques for assessment emerge it is helpful to have clear definitions of these notions. This memo provides more detailed definitions, defines a new category for combinations of traditional active and passive techniques, and discusses dimensions to evaluate new techniques as they emerge.

This memo provides definitions for Active and Passive Metrics and Methods based on long usage in the Internet measurement community, and especially the Internet Engineering Task Force. This memo also describes the combination of fundamental Active and Passive categories, which are called Hybrid Methods and Metrics.

1.1. Requirements Language

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

2. Purpose and Scope

The scope of this memo is to define and describe Active and Passive versions of metrics and methods which are consistent with the long-time usage of these adjectives in the Internet measurement community and especially the Internet Engineering Task Force. Since the science of measurement is expanding, we provide a category for combinations of the traditional extremes, treating Active and Passive as a continuum and designating combinations of their attributes as Hybrid methods.

Further, this memo's purpose includes describing multiple dimensions to evaluate new methods as they emerge.

3. Terms and Definitions

This section defines the key terms of the memo. Some definitions use the notion of "stream of interest" which is synonymous with "population of interest" defined in clause 6.1.1 of ITU-T Recommendation Y.1540 [Y.1540]. These definitions will be useful for work-in-progress, such as [I-D.zheng-ippm-framework-passive] (with which there is already good consistency).

3.1. Performance Metric

The standard definition of a quantity, produced in an assessment of performance and/or reliability of the network, which has an intended utility and is carefully specified to convey the exact meaning of a measured value. (This definition is consistent with that of Performance Metric in [RFC2330] and [RFC6390]).

3.2. Method of Measurement

The procedure or set of operations having the object of determining a Measured Value or Measurement Result.

3.3. Observation Point

See section 2 of [RFC7011] for this definition (a location in the network where packets can be observed), and related definitions. The comparable term defined in IETF literature on Active measurement is Measurement Point, see section 4.1 of [RFC5835]. Both of these terms have come into use describing similar actions at the identified point in the network path.

3.4. Active Methods

Active measurement methods have the following attributes:

1. Active methods generate packet streams. Commonly, the packet stream of interest is generated as the basis of measurement. Sometimes, the adjective "synthetic" is used to categorize Active measurement streams [Y.1731]. Accompanying packet stream(s) may be generated to increase overall traffic load, though the loading stream(s) may not be measured.
2. The packets in the stream of interest have fields or field values (or are augmented or modified to include fields or field values) which are dedicated to measurement. Since measurement usually requires determining the corresponding packets at multiple measurement points, a sequence number is the most common information dedicated to measurement, and often combined with a timestamp.
3. The Source and Destination of the packet stream of interest are usually known a priori.
4. The characteristics of the packet stream of interest are known at the Source at least, and may be communicated to Destination as part of the method. Note that some packet characteristics will

normally change during packet forwarding. Other changes along the path are possible, see [I-D.morton-ippm-2330-stdform-typep].

When adding traffic to the network for measurement, Active Methods influence the quantities measured to some degree, and those performing tests should take steps to quantify the effect(s) and/or minimize such effects.

3.5. Active Metric

An Active Metric incorporates one or more of the aspects of Active Methods in the metric definition.

For example, IETF metrics for IP performance (developed according to the [RFC2330] framework) include the Source packet stream characteristics as metric input parameters, and also specify the packet characteristics (Type-P) and Source and Destination IP addresses (with their implications on both stream treatment and interfaces associated with measurement points).

3.6. Passive Methods

Passive measurement methods are

- o based solely on observations of undisturbed and unmodified packet stream of interest (in other words, the method of measurement MUST NOT add, change, or remove packets or fields, or change field values anywhere along the path).
- o dependent on the existence of one or more packet streams to supply the stream of interest.
- o dependent on the presence of the packet stream of interest at one or more designated observation points.

Some passive methods simply observe and collect information on all packets that pass Observation Point(s), while others filter the packets as a first step and only collect information on packets that match the filter criteria, and thereby narrow the stream of interest.

It is common that passive methods are conducted at one or more Observation Points. Passive methods to assess Performance Metrics often require multiple observation points, e.g., to assess latency of packet transfer across a network path between two Observation Points. In this case, the observed packets must include enough information to determine the corresponding packets at different Observation Points.

Communication of the observations (in some form) to a collector is an essential aspect of Passive Methods. In some configurations, the traffic load associated with results export to a collector may influence the network performance. However, the collection of results is not unique to Passive Methods, and the load from management and operations of measurement systems must always be considered for potential effects on the measured values.

3.7. Passive Metric

Passive Metrics apply to observations of packet traffic (traffic flows in [RFC7011]).

Passive performance metrics are assessed independent of the packets or traffic flows, and solely through observation. Some refer to such assessments as "out-of-band".

One example of passive performance metrics for IP packet transfer can be found in ITU-T Recommendation Y.1540 [Y.1540], where the metrics are defined on the basis of reference events generated as packet pass reference points. The metrics are agnostic to the distinction between active and passive when the necessary packet correspondence can be derived from the observed stream of interest as required.

3.8. Hybrid Methods and Metrics

Hybrid Methods are Methods of Measurement which 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. ITU-T Recommendation Y.1540 [Y.1540] defines metrics that are also applicable to the hybrid categories, since packet correspondence at different observation/reference points could be derived from "fields or field values which are dedicated to measurement", but otherwise the methods are passive.

There are several types of Hybrid methods, as categorized below.

With respect to a *single* stream of interest, Hybrid Type I methods fit in the continuum as follows, in terms of what happens at the Source (or Observation Point nearby):

- o If you generate the stream of interest => Active
- o If you augment or modify the stream of interest, or employ methods that modify the treatment of the stream => Hybrid Type I
- o If you solely observe a stream of interest => Passive

As an example, consider the case where the method generates traffic load stream(s), and observes an existing stream of interest according to the criteria for Passive Methods. Since loading streams are an aspect of Active Methods, the stream of interest is not "solely observed", and the measurements involve a single stream of interest whose treatment has been modified both the presence of the load. Therefore, this is a Hybrid Type I method.

We define Hybrid Type II as follows: Methods that employ two or more different streams of interest with some degree of mutual coordination (e.g., one or more Active streams and one or more undisturbed and unmodified packet streams) to collect both Active and Passive Metrics and enable enhanced characterization from additional joint analysis. [I-D.trammell-ippm-hybrid-ps] presents a problem statement for Hybrid Type II methods and metrics. Note that one or more Hybrid Type I streams could be substituted for the Active streams or undisturbed streams in the mutually coordinated set. It is the Type II Methods where unique Hybrid Metrics are anticipated to emerge.

Methods based on a combination of a single (generated) Active stream and Passive observations applied to the stream of interest at intermediate observation points are also a type of Hybrid Methods. However, [RFC5644] already defines these as Spatial Metrics and Methods. It is possible to replace the Active stream of [RFC5644] with a Hybrid Type I stream and measure Spatial Metrics (but this was un-anticipated when [RFC5644] was developed).

The Table below illustrates the categorization of methods (where "Synthesis" refers to a combination of Active and Passive Method attributes).

	Single Stream of Interest	Multiple Simultaneous Streams of Interest from Different Methods
Single Fundamental Method	Active or Passive	
Synthesis of Fundamental Methods	Hybrid Type I	
Multiple Methods	Spatial Metrics [RFC5644]	Hybrid Type II

There may be circumstances where results measured with Hybrid Methods can be considered equivalent to Passive Methods. Referencing the notion of a "class C" where packets of different Type-P are treated

equally in Section 13 of [RFC2330] and the terminology for paths from Section 5 of [RFC2330]:

Hybrid Methods of Measurement that augment or modify packets of a "class C" in a host should produce equivalent results to Passive Methods of Measurement, when hosts accessing and links transporting these packets along the path (other than those performing augmentation/modification) treat packets from both categories of methods (with and without the augmentation/modification) as the same "class C". The Passive Methods of Measurement represent the Ground Truth for comparisons of results between Passive and Hybrid methods, and this comparison should be conducted to confirm the class C treatment.

4. Discussion

This section illustrates the definitions and presents some examples.

4.1. Graphical Representation

If we compare the Active and Passive Methods, there are at least two dimensions on which methods can be evaluated. This evaluation space may be useful when a method is a combination of the two alternative methods.

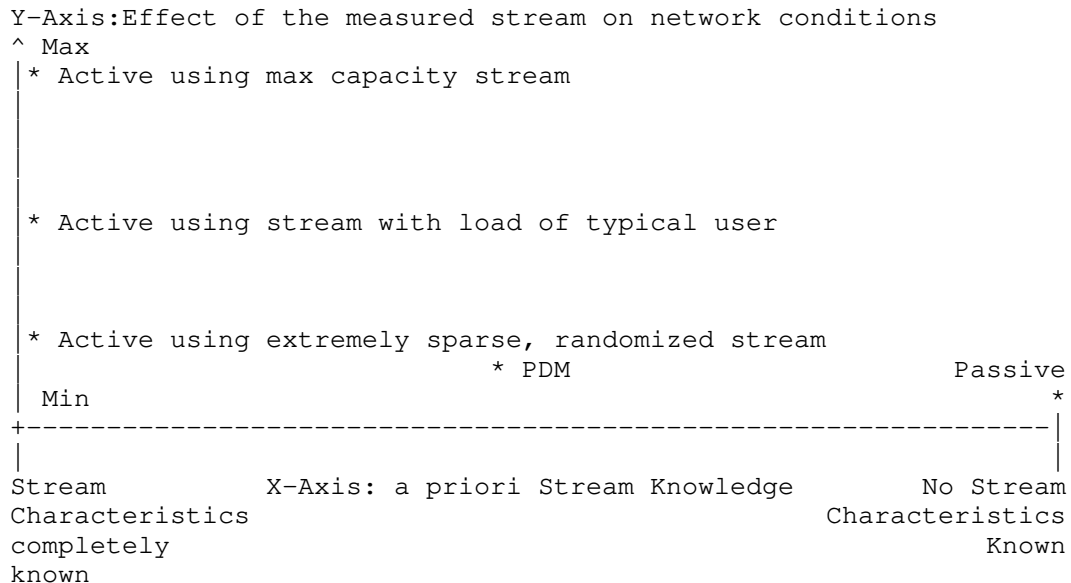
The two dimensions (initially chosen) are:

Y-Axis: "Effect of the measured stream on network conditions." The degree to which the stream of interest biases overall network conditions experienced by that stream and other streams. This is a key dimension for Active measurement error analysis. (Comment: There is also the notion of time averages - a measurement stream may have significant effect while it is present, but the stream is only generated 0.1% of the time. On the other hand, observations alone have no effect on network performance. To keep these dimensions simple, we consider the stream effect only when it is present, but note that reactive networks defined in [RFC7312] may exhibit bias for some time beyond the life of a stream.)

X-Axis: "a priori Stream Knowledge." The degree to which stream characteristics are known a priori. There are methodological advantages of knowing the source stream characteristics, and having complete control of the stream characteristics. For example, knowing the number of packets in a stream allows more efficient operation of the measurement receiver, and so is an asset for active measurement methods. Passive methods (with no sample filter) have few clues available to anticipate what the protocol first packet observed will use or how many packets will

comprise the flow, but once the standard protocol of a flow is known the possibilities narrow (for some compliant flows). Therefore this is a key dimension for Passive measurement error analysis.

There are a few examples we can plot on a two-dimensional space. We can anchor the dimensions with reference point descriptions.



(In the graph above, "PDM" refers to [I-D.ietf-ippm-6man-pdm-option], an IPv6 Option Header for Performance and Diagnostic Measurements, described in section 4.2.)

We recognize that method categorization could be based on additional dimensions, but this would require a different graphical approach.

For example, "effect of stream of interest on network conditions" could easily be further qualified into:

1. effect on the performance of the stream of interest itself: for example, choosing a packet marking or Differentiated Services Code Point (DSCP) resulting in domain treatment as a real-time stream (as opposed to default/best-effort marking).

2. effect on unmeasured streams that share the path and/or bottlenecks: for example, an extremely sparse measured stream of minimal size packets typically has little effect on other flows (and itself), while a stream designed to characterize path capacity may affect all other flows passing through the capacity bottleneck (including itself).
3. effect on network conditions resulting in network adaptation: for example, a network monitoring load and congestion conditions might change routing, placing some flows to alternate paths to mitigate the congestion.

We have combined 1 and 2 on the Y-axis, as examination of examples indicates strong correlation of effects in this pair, and network adaptation is not addressed.

It is apparent that different methods of IP network measurement can produce different results, even when measuring the same path at the same time. The two dimensions of the graph help to understand how the results might change with the method chosen. For example, an Active Method to assess throughput adds some amount of traffic to the network which might result in lower throughput for all streams. However, a Passive Method to assess throughput can also err on the low side due to unknown limitations of the hosts providing traffic, competition for host resources, limitations of the network interface, or private sub-networks that are not an intentional part of the path, etc. And Hybrid Methods could easily suffer from both forms of error. Another example of potential errors stems from the pitfalls of using an Active stream with known bias, such as a periodic stream defined in [RFC3432]. The strength of modelling periodic streams (like VoIP) is a potential weakness when extending the measured results to other application whose streams are non-periodic. The solutions are to model the application streams more exactly with an Active Method, or accept the risks and potential errors with the Passive Method discussed above.

4.2. Discussion of PDM

In [I-D.ietf-ippm-6man-pdm-option], an IPv6 Option Header for Performance and Diagnostic Measurements (PDM) is described which (when added to the stream of interest at strategic interfaces) supports performance measurements. This method processes a user traffic stream and adds "fields which are dedicated to measurement" (the measurement intent is made clear in the title of this option). Thus:

- o The method intends to have a small effect on the measured stream and other streams in the network. There are conditions where this intent may not be realized.
- o The measured stream has unknown characteristics until it is processed to add the PDM Option header. Note that if the packet MTU is exceeded after adding the header, the intent to have small effect will not be realized.

We conclude that this is a Hybrid Type I method, having at least one characteristic of both active and passive methods for a single stream of interest.

4.3. Discussion of "Coloring" Method

Draft [I-D.tempi-opsawg-p3m], proposed to color packets by re-writing a field of the stream at strategic interfaces to support performance measurements (noting that this is a difficult operation at an intermediate point on an encrypted Virtual Private Network). This method processes a user traffic stream and inserts "fields or values which are dedicated to measurement". Thus:

- o The method intends to have a small effect on the measured stream and other streams in the network (smaller than PDM above). There are conditions where this intent may not be realized.
- o The measured stream has unknown characteristics until it is processed to add the coloring in the header, and the stream could be measured and time-stamped during that process.

We note that [I-D.chen-ippm-coloring-based-ipfpm-framework] proposes a method similar to [I-D.tempi-opsawg-p3m], as ippm-list discussion revealed.

We conclude that this is a Hybrid Type I method, having at least one characteristic of both active and passive methods for a single stream of interest.

4.4. Brief Discussion of OAM Methods

Many Operations, Administration, and Management (OAM) methods exist beyond the IP-layer. For example, [Y.1731] defines several different measurement methods which we would classify as follows:

- o Loss Measurement (LM) occasionally injects frames with a count of previous frames since the last LM message. We conclude LM is Hybrid Type I because

- A. This method processes a user traffic stream,
 - B. and augments the stream of interest with frames having "fields which are dedicated to measurement".
- o Synthetic Loss Measurement (SLM) and Delay Measurement (DM) methods both inject dedicated measurement frames, so the "stream of interest is generated as the basis of measurement". We conclude that SLM and DM methods are Active Methods.

We also recognize the existence of alternate terminology used in OAM at layers other than IP. Readers are encouraged to consult [RFC6374] for MPLS Loss and Delay measurement terminology, for example.

5. Security considerations

When considering security and privacy of those involved in measurement or those whose traffic is measured, there is sensitive information communicated and observed at observation and measurement points described above, and protocol issues to consider. We refer the reader to the security and privacy considerations described in the Large Scale Measurement of Broadband Performance (LMAP) Framework [RFC7594], which covers active and passive measurement techniques and supporting material on measurement context.

6. IANA Considerations

This memo makes no requests for IANA consideration.

7. Acknowledgements

Thanks to Mike Ackermann for asking the right question, and for several suggestions on terminology. Brian Trammell provided key terms and references for the passive category, and suggested ways to expand the Hybrid description and types. Phil Eardley suggested some hybrid scenarios for categorization as part of his review. Tiziano Ionta reviewed the draft and suggested the classification for the "coloring" method of measurement. Nalini Elkins identified several areas for clarification following her review. Bill Jouris, Stenio Fernandes, and Spencer Dawkins suggested several editorial improvements. Tal Mizrahi, Joachim Fabini, Greg Mirsky and Mike Ackermann raised many key considerations in their WGLC reviews, based on their broad measurement experience.

8. References

8.1. Normative References

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

- [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>>.

8.2. Informative References

- [RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <<http://www.rfc-editor.org/info/rfc6374>>.
- [I-D.morton-ippm-2330-stdform-typep]
Morton, A., Fabini, J., Elkins, N., mackermann@bcbsm.com, m., and V. Hegde, "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 (work in progress), December 2015.
- [I-D.ietf-ippm-6man-pdm-option]
Elkins, N. and M. Ackermann, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", draft-ietf-ippm-6man-pdm-option-01 (work in progress), October 2015.
- [I-D.tempia-opsawg-p3m]
Capello, A., Cociglio, M., Castaldelli, L., and A. Bonda, "A packet based method for passive performance monitoring", draft-tempia-opsawg-p3m-04 (work in progress), February 2014.
- [I-D.chen-ippm-coloring-based-ipfpm-framework]
Chen, M., Zheng, L., Mirsky, G., and G. Fioccola, "IP Flow Performance Measurement Framework", draft-chen-ippm-coloring-based-ipfpm-framework-04 (work in progress), July 2015.
- [I-D.zheng-ippm-framework-passive]
Zheng, L., Elkins, N., Lingli, D., Ackermann, M., and G. Mirsky, "Framework for IP Passive Performance Measurements", draft-zheng-ippm-framework-passive-03 (work in progress), February 2015.
- [I-D.trammell-ippm-hybrid-ps]
Trammell, B., Zheng, L., Berenguer, S., and M. Bagnulo, "Hybrid Measurement using IPPM Metrics", draft-trammell-ippm-hybrid-ps-01 (work in progress), February 2014.

- [Y.1540] ITU-T Recommendation Y.1540, , "Internet protocol data communication service - IP packet transfer and availability performance parameters", March 2011.
- [Y.1731] ITU-T Recommendation Y.1731, , "Operation, administration and management (OAM) functions and mechanisms for Ethernet-based networks", October 2015.

Author's Address

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

Email: acmorton@att.com

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

M. Bagnulo
UC3M
B. Claise
Cisco Systems, Inc.
P. Eardley
BT
A. Morton
AT&T Labs
A. Akhter
Consultant
March 9, 2020

Registry for Performance Metrics
draft-ietf-ippm-metric-registry-24

Abstract

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

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 10, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Terminology	5
3. Scope	7
4. Motivation for a Performance Metrics Registry	8
4.1. Interoperability	8
4.2. Single point of reference for Performance Metrics	9
4.3. Side benefits	9
5. Criteria for Performance Metrics Registration	9
6. Performance Metric Registry: Prior attempt	10
6.1. Why this Attempt Should Succeed	11
7. Definition of the Performance Metric Registry	11
7.1. Summary Category	13
7.1.1. Identifier	13
7.1.2. Name	13
7.1.3. URI	17
7.1.4. Description	17
7.1.5. Reference	17
7.1.6. Change Controller	17
7.1.7. Version (of Registry Format)	18
7.2. Metric Definition Category	18
7.2.1. Reference Definition	18
7.2.2. Fixed Parameters	18
7.3. Method of Measurement Category	19
7.3.1. Reference Method	19
7.3.2. Packet Stream Generation	19
7.3.3. Traffic Filter	20
7.3.4. Sampling Distribution	20
7.3.5. Run-time Parameters	21
7.3.6. Role	22
7.4. Output Category	22
7.4.1. Type	22
7.4.2. Reference Definition	23
7.4.3. Metric Units	23
7.4.4. Calibration	23
7.5. Administrative information	24
7.5.1. Status	24
7.5.2. Requester	24
7.5.3. Revision	24
7.5.4. Revision Date	24
7.6. Comments and Remarks	24

8. Processes for Managing the Performance Metric Registry Group	24
8.1. Adding new Performance Metrics to the Performance Metrics Registry	25
8.2. Revising Registered Performance Metrics	26
8.3. Deprecating Registered Performance Metrics	28
9. Security considerations	28
10. IANA Considerations	29
10.1. Registry Group	29
10.2. Performance Metric Name Elements	29
10.3. New Performance Metrics Registry	30
11. Blank Registry Template	32
11.1. Summary	32
11.1.1. ID (Identifier)	32
11.1.2. Name	32
11.1.3. URI	32
11.1.4. Description	32
11.1.5. Change Controller	32
11.1.6. Version (of Registry Format)	32
11.2. Metric Definition	32
11.2.1. Reference Definition	32
11.2.2. Fixed Parameters	32
11.3. Method of Measurement	33
11.3.1. Reference Method	33
11.3.2. Packet Stream Generation	33
11.3.3. Traffic Filtering (observation) Details	33
11.3.4. Sampling Distribution	33
11.3.5. Run-time Parameters and Data Format	33
11.3.6. Roles	33
11.4. Output	33
11.4.1. Type	34
11.4.2. Reference Definition	34
11.4.3. Metric Units	34
11.4.4. Calibration	34
11.5. Administrative items	34
11.5.1. Status	34
11.5.2. Requester	34
11.5.3. Revision	34
11.5.4. Revision Date	34
11.6. Comments and Remarks	34
12. Acknowledgments	34
13. References	35
13.1. Normative References	35
13.2. Informative References	36
Authors' Addresses	37

1. Introduction

The IETF specifies and uses Performance Metrics of protocols and applications transported over its protocols. Performance metrics are important part of network operations using IETF protocols, and [RFC6390] specifies guidelines for their development.

The definition and use of Performance Metrics in the IETF has been fostered in various working groups (WG), most notably:

The "IP Performance Metrics" (IPPM) WG is the WG primarily focusing on Performance Metrics definition at the IETF.

The "Benchmarking Methodology" WG (BMWG) defines many Performance Metrics for use in laboratory benchmarking of inter-networking technologies.

The "Metric Blocks for use with RTCP's Extended Report Framework" (XRBLOCK) WG (concluded) specified many Performance Metrics related to "RTP Control Protocol Extended Reports (RTCP XR)" [RFC3611], which establishes a framework to allow new information to be conveyed in RTCP, supplementing the original report blocks defined in "RTP: A Transport Protocol for Real-Time Applications", [RFC3550].

The "IP Flow Information eXport" (IPFIX) concluded WG specified an IANA process for new Information Elements. Some Performance Metrics related Information Elements are proposed on regular basis.

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

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

Despite the importance of Performance Metrics, there are two related problems for the industry. First, ensuring that when one party requests another party to measure (or report or in some way act on) a particular Performance Metric, then both parties have exactly the same understanding of what Performance Metric is being referred to. Second, discovering which Performance Metrics have been specified, to avoid developing a new Performance Metric that is very similar, but not quite inter-operable. These problems can be addressed by creating a registry of performance metrics. The usual way in which the IETF organizes registries is with Internet Assigned Numbers

Authority (IANA), and there is currently no Performance Metrics Registry maintained by the IANA.

This document requests that IANA create and maintain a Performance Metrics Registry, according to the maintenance procedures and the Performance Metrics Registry format defined in this memo. The resulting Performance Metrics Registry is for use by the IETF and others. Although the Registry formatting specifications herein are primarily for registry creation by IANA, any other organization that wishes to create a performance metrics registry may use the same formatting specifications for their purposes. The authors make no guarantee of the registry format's applicability to any possible set of Performance Metrics envisaged by other organizations, but encourage others to apply it. In the remainder of this document, unless we explicitly say otherwise, we will refer to the IANA-maintained Performance Metrics Registry as simply the Performance Metrics Registry.

2. Terminology

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

Performance Metric: A Performance Metric is a quantitative measure of performance, targeted to an IETF-specified protocol or targeted to an application transported over an IETF-specified protocol. Examples of Performance Metrics are the FTP response time for a complete file download, the DNS response time to resolve the IP address(es), a database logging time, etc. This definition is consistent with the definition of metric in [RFC2330] and broader than the definition of performance metric in [RFC6390].

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

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

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

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

Parameter: A Parameter is an input factor defined as a variable in the definition of a Performance Metric. A Parameter is a numerical or other specified factor forming one of a set that defines a metric or sets the conditions of its operation. All Parameters must be known in order to make a measurement using a metric and interpret the results. There are two types of Parameters: Fixed and Run-time parameters. For the Fixed Parameters, the value of the variable is specified in the Performance Metrics Registry entry and different Fixed Parameter values results in different Registered Performance Metrics. For the Run-time Parameters, the value of the variable is defined when the metric measurement method is executed and a given Registered Performance Metric supports multiple values for the parameter. Although Run-time Parameters do not change the fundamental nature of the Performance Metric's definition, some have substantial influence on the network property being assessed and interpretation of the results.

Note: Consider the case of packet loss in the following two Active Measurement Method cases. The first case is packet loss as background loss where the Run-time Parameter set includes a very sparse Poisson stream, and only characterizes the times when packets were lost. Actual user streams likely see much higher loss at these times, due to tail drop or radio errors. The second case is packet loss as inverse of throughput where the Run-time Parameter set includes a very dense, bursty stream, and characterizes the loss experienced by a stream that approximates a user stream. These are both "loss metrics", but the difference in interpretation of the results is highly dependent on the Run-time Parameters (at least), to the extreme where we are actually using loss to infer its compliment: delivered throughput.

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

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

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

3. Scope

This document is intended for two different audiences:

1. For those defining new Registered Performance Metrics, it provides specifications and best practices to be used in deciding which Registered Performance Metrics are useful for a measurement study, instructions for writing the text for each column of the Registered Performance Metrics, and information on the supporting documentation required for the new Performance Metrics Registry entry (up to and including the publication of one or more immutable documents such as an RFC).
2. For the appointed Performance Metrics Experts and for IANA personnel administering the new IANA Performance Metrics Registry, it defines a set of acceptance criteria against which these proposed Registered Performance Metrics should be evaluated.

In addition, this document may be useful for other organizations who are defining a Performance Metric registry of their own, and may re-use the features of the Performance Metrics Registry defined in this document.

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

document gives a set of guidelines for requesters and expert reviewers of candidate Registered Performance Metrics.

This document makes no attempt to populate the Performance Metrics Registry with initial entries; the related memo [I-D.ietf-ippm-initial-registry] proposes the initial set of registry entries.

4. Motivation for a Performance Metrics Registry

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

4.1. Interoperability

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

- o Control protocol: This type of protocol used to allow one entity to request another entity to perform a measurement using a specific metric defined by the Performance Metrics Registry. One particular example is the LMAP framework [RFC7594]. Using the LMAP terminology, the Performance Metrics Registry is used in the LMAP Control protocol to allow a Controller to schedule a measurement task for one or more Measurement Agents. In order to enable this use case, the entries of the Performance Metrics Registry must be sufficiently defined to allow a Measurement Agent implementation to trigger a specific measurement task upon the reception of a control protocol message. This requirement heavily constrains the type of entries that are acceptable for the Performance Metrics Registry.
- o Report protocol: This type of protocol is used to allow an entity to report measurement results to another entity. By referencing to a specific Performance Metrics Registry, it is possible to properly characterize the measurement result data being reported. Using the LMAP terminology, the Performance Metrics Registry is used in the Report protocol to allow a Measurement Agent to report measurement results to a Collector.

It should be noted that the LMAP framework explicitly allows for using not only the IANA-maintained Performance Metrics Registry but also other registries containing Performance Metrics, either defined by other organizations or private ones. However, others who are

creating Registries to be used in the context of an LMAP framework are encouraged to use the Registry format defined in this document, because this makes it easier for developers of LMAP Measurement Agents (MAs) to programmatically use information found in those other Registries' entries.

4.2. Single point of reference for Performance Metrics

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

4.3. Side benefits

There are a couple of side benefits of having such a registry. First, the Performance Metrics Registry could serve as an inventory of useful and used Performance Metrics, that are normally supported by different implementations of measurement agents. Second, the results of measurements using the Performance Metrics should be comparable even if they are performed by different implementations and in different networks, as the Performance Metric is properly defined. BCP 176 [RFC6576] examines whether the results produced by independent implementations are equivalent in the context of evaluating the completeness and clarity of metric specifications. This BCP defines the standards track advancement testing for (active) IPPM metrics, and the same process will likely suffice to determine whether Registered Performance Metrics are sufficiently well specified to result in comparable (or equivalent) results. Registered Performance Metrics which have undergone such testing SHOULD be noted, with a reference to the test results.

5. Criteria for Performance Metrics Registration

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

1. interpretable by the user.
2. implementable by the software or hardware designer,
3. deployable by network operators,
4. accurate in terms of producing equivalent results, and for interoperability and deployment across vendors,
5. Operationally useful, so that it has significant industry interest and/or has seen deployment,
6. Sufficiently tightly defined, so that different values for the Run-time Parameters does not change the fundamental nature of the measurement, nor change the practicality of its implementation.

In essence, there needs to be evidence that a candidate Registered Performance Metric has significant industry interest, or has seen deployment, and there is agreement that the candidate Registered Performance Metric serves its intended purpose.

6. Performance Metric Registry: Prior attempt

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

A couple of interesting additional quotes from RFC 6248 [RFC6248] might help to understand the issues related to that registry.

1. "It is not believed to be feasible or even useful to register every possible combination of Type P, metric parameters, and Stream parameters using the current structure of the IPPM Metrics Registry."
2. "The registry structure has been found to be insufficiently detailed to uniquely identify IPPM metrics."
3. "Despite apparent efforts to find current or even future users, no one responded to the call for interest in the RFC 4148 registry during the second half of 2010."

The current approach learns from this by tightly defining each Registered Performance Metric with only a few variable (Run-time) Parameters to be specified by the measurement designer, if any. The

idea is that entries in the Performance Metrics Registry stem from different measurement methods which require input (Run-time) parameters to set factors like source and destination addresses (which do not change the fundamental nature of the measurement). The downside of this approach is that it could result in a large number of entries in the Performance Metrics Registry. There is agreement that less is more in this context - it is better to have a reduced set of useful metrics rather than a large set of metrics, some with questionable usefulness.

6.1. Why this Attempt Should Succeed

As mentioned in the previous section, one of the main issues with the previous registry was that the metrics contained in the registry were too generic to be useful. This document specifies stricter criteria for performance metric registration (see section 5), and imposes a group of Performance Metrics Experts that will provide guidelines to assess if a Performance Metric is properly specified.

Another key difference between this attempt and the previous one is that in this case there is at least one clear user for the Performance Metrics Registry: the LMAP framework and protocol. Because the LMAP protocol will use the Performance Metrics Registry values in its operation, this actually helps to determine if a metric is properly defined. In particular, since we expect that the LMAP control protocol will enable a controller to request a measurement agent to perform a measurement using a given metric by embedding the Performance Metrics Registry identifier in the protocol. Such a metric and method are properly specified if they are defined well-enough so that it is possible (and practical) to implement them in the measurement agent. This was the failure of the previous attempt: a registry entry with an undefined Type-P (section 13 of RFC 2330 [RFC2330]) allows implementation to be ambiguous.

7. Definition of the Performance Metric Registry

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

column(s) MUST give general guidelines for populating the new column(s) for existing entries.

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

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

=====

Legend:

Registry Categories and Columns are shown below as:

Category

-----...

Column | Column |...

=====

Summary

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

Metric Definition

Reference Definition	Fixed Parameters
----------------------	------------------

Method of Measurement

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

Output

Type	Reference Definition	Units	Calibration
------	----------------------	-------	-------------

Administrative Information

Status	Requester	Rev	Rev.Date
--------	-----------	-----	----------

Comments and Remarks

There is a blank template of the Registry template provided in Section 11 of this memo.

7.1. Summary Category

7.1.1. Identifier

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

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

The Identifier 0 should be Reserved. The Identifier values from 64512 to 65536 are reserved for private or experimental use, and the user may encounter overlapping uses.

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

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

7.1.2. Name

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

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

MetricType_Method_SubTypeMethod_... Spec_Units_Output

- o MetricType: a combination of the directional properties and the metric measured, such as and not limited to:

- RTDelay (Round Trip Delay)

- RTDNS (Response Time Domain Name Service)

- RLDNS (Response Loss Domain Name Service)

- OWDelay (One Way Delay)

RTLoss (Round Trip Loss)

OWLoss (One Way Loss)

OWPDV (One Way Packet Delay Variation)

OWIPDV (One Way Inter-Packet Delay Variation)

OWReorder (One Way Packet Reordering)

OWDuplic (One Way Packet Duplication)

OWBTC (One Way Bulk Transport Capacity)

OWMBM (One Way Model Based Metric)

SPMonitor (Single Point Monitor)

MPMonitor (Multi-Point Monitor)

- o Method: One of the methods defined in [RFC7799], such as and not limited to:

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

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

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

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

Spatial (Spatial Metric of RFC5644)

- o SubTypeMethod: One or more sub-types to further describe the features of the entry, such as and not limited to:

ICMP (Internet Control Message Protocol)

IP (Internet Protocol)

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

UDP (User Datagram Protocol)

TCP (Transport Control Protocol)

QUIC (QUIC transport protocol)

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

Poisson (Packet generation using Poisson distribution)

Periodic (Periodic packet generation)

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

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

SustainedBurst (Capacity test, worst case)

StandingQueue (test of bottleneck queue behavior)

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

- o Spec: An immutable document identifier combined with a document section identifier. For RFCs, this consists of the RFC number and major section number that specifies this Registry entry in the form RFCXXXXsecY, such as RFC7799sec3. Note: the RFC number is not the Primary Reference specification for the metric definition, such as [RFC7679] for One-way Delay; it will contain the placeholder "RFCXXXXsecY" until the RFC number is assigned to the specifying document, and would remain blank in private registry entries without a corresponding RFC. Anticipating the "RFC10K" problem, the number of the RFC continues to replace RFCXXXX regardless of the number of digits in the RFC number. Anticipating Registry Entries from other standards bodies, the form of this Name Element MUST be proposed and reviewed for consistency and uniqueness by the Expert Reviewer.
- o Units: The units of measurement for the output, such as and not limited to:

Seconds

Ratio (unitless)

Percent (value multiplied by 100%)

Logical (1 or 0)

Packets

BPS (Bits per Second)

PPS (Packets per Second)

EventTotal (for unit-less counts)

Multiple (more than one type of unit)

Enumerated (a list of outcomes)

Unitless

- o Output: The type of output resulting from measurement, such as and not limited to:

Singleton

Raw (multiple Singletons)

Count

Minimum

Maximum

Median

Mean

95Percentile (95th Percentile)

99Percentile (99th Percentile)

StdDev (Standard Deviation)

Variance

PFI (Pass, Fail, Inconclusive)

FlowRecords (descriptions of flows observed)

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

An example is:

RTDelay_Active_IP-UDP-Periodic_RFCXXXXsecY_Seconds_95Percentile

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

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

7.1.3. URI

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

7.1.4. Description

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

7.1.5. Reference

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

7.1.6. Change Controller

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

7.1.7. Version (of Registry Format)

This entry gives the version number for the registry format used. Formats complying with this memo MUST use 1.0. The version number SHALL NOT change unless a new RFC is published that changes the registry format. The version number of registry entries SHALL NOT change unless the registry entry is updated (following procedures in section 8).

7.2. Metric Definition Category

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

7.2.1. Reference Definition

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

7.2.2. Fixed Parameters

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

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

Parameters MUST have well-defined names. For human readers, the hanging indent style is preferred, and any Parameter names and

definitions that do not appear in the Reference Method Specification MUST appear in this column (or Run-time Parameters column).

Parameters MUST have a well-specified data format.

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

7.3. Method of Measurement Category

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

7.3.1. Reference Method

This entry provides references to relevant sections of immutable documents, such as RFC(s) (for other standards bodies, it is likely to be necessary to reference a specific, dated version of a specification) describing the method of measurement, as well as any supplemental information needed to ensure unambiguous interpretation for implementations referring to the immutable document text.

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

7.3.2. Packet Stream Generation

This column applies to Performance Metrics that generate traffic as part of their Measurement Method, including but not necessarily limited to Active metrics. The generated traffic is referred as a stream and this column describes its characteristics.

Each entry for this column contains the following information:

- o Value: The name of the packet stream scheduling discipline
- o Reference: the specification where the parameters of the stream are defined

The packet generation stream may require parameters such as the average packet rate and distribution truncation value for streams with Poisson-distributed inter-packet sending times. In case such parameters are needed, they should be included either in the Fixed parameter column or in the run time parameter column, depending on whether they will be fixed or will be an input for the metric.

The simplest example of stream specification is Singleton scheduling (see [RFC2330]), where a single atomic measurement is conducted. Each atomic measurement could consist of sending a single packet (such as a DNS request) or sending several packets (for example, to request a webpage). Other streams support a series of atomic measurements in a "sample", with a schedule defining the timing between each transmitted packet and subsequent measurement. Principally, two different streams are used in IPPM metrics, Poisson distributed as described in [RFC2330] and Periodic as described in [RFC3432]. Both Poisson and Periodic have their own unique parameters, and the relevant set of parameters names and values should be included either in the Fixed Parameters column or in the Run-time parameter column.

7.3.3. Traffic Filter

This column applies to Performance Metrics that observe packets flowing through (the device with) the measurement agent i.e. that is not necessarily addressed to the measurement agent. This includes but is not limited to Passive Metrics. The filter specifies the traffic that is measured. This includes protocol field values/ranges, such as address ranges, and flow or session identifiers.

The traffic filter itself depends on needs of the metric itself and a balance of an operator's measurement needs and a user's need for privacy. Mechanics for conveying the filter criteria might be the BPF (Berkley Packet Filter) or PSAMP [RFC5475] Property Match Filtering which reuses IPFIX [RFC7012]. An example BPF string for matching TCP/80 traffic to remote destination net 192.0.2.0/24 would be "dst net 192.0.2.0/24 and tcp dst port 80". More complex filter engines might be supported by the implementation that might allow for matching using Deep Packet Inspection (DPI) technology.

The traffic filter includes the following information:

Type: the type of traffic filter used, e.g. BPF, PSAMP, OpenFlow rule, etc. as defined by a normative reference

Value: the actual set of rules expressed

7.3.4. Sampling Distribution

The sampling distribution defines out of all the packets that match the traffic filter, which one of those are actually used for the measurement. One possibility is "all" which implies that all packets matching the Traffic filter are considered, but there may be other sampling strategies. It includes the following information:

Value: the name of the sampling distribution

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

The sampling distribution may require parameters. In case such parameters are needed, they should be included either in the Fixed parameter column or in the run time parameter column, depending on whether they will be fixed or will be an input for the metric.

Sampling and Filtering Techniques for IP Packet Selection are documented in the PSAMP (Packet Sampling) [RFC5475], while the Framework for Packet Selection and Reporting, [RFC5474] provides more background information. The sampling distribution parameters might be expressed in terms of the Information Model for Packet Sampling Exports, [RFC5477], and the Flow Selection Techniques, [RFC7014].

7.3.5. Run-time Parameters

Run-Time Parameters are Parameters that must be determined, configured into the measurement system, and reported with the results for the context to be complete. However, the values of these parameters is not specified in the Performance Metrics Registry (like the Fixed Parameters), rather these parameters are listed as an aid to the measurement system implementer or user (they must be left as variables, and supplied on execution).

Where metrics supply a list of Parameters as part of their descriptive template, a sub-set of the Parameters will be designated as Run-Time Parameters.

Parameters MUST have well defined names. For human readers, the hanging indent style is preferred, and the names and definitions that do not appear in the Reference Method Specification MUST appear in this column.

A Data Format for each Run-time Parameter MUST be specified in this column, to simplify the control and implementation of measurement devices. For example, parameters that include an IPv4 address can be encoded as a 32 bit integer (i.e. binary base64 encoded value) or ip-address as defined in [RFC6991]. The actual encoding(s) used must be explicitly defined for each Run-time parameter. IPv6 addresses and options MUST be accommodated, allowing Registered Metrics to be used in that address family. Other address families are permissable.

Examples of Run-time Parameters include IP addresses, measurement point designations, start times and end times for measurement, and other information essential to the method of measurement.

7.3.6. Role

In some methods of measurement, there may be several roles defined, e.g., for a one-way packet delay active measurement there is one measurement agent that generates the packets and another agent that receives the packets. This column contains the name of the Role(s) for this particular entry. In the one-way delay example above, there should be two entries in the Role registry column, one for each Role (Source and Destination). When a measurement agent is instructed to perform the "Source" Role for one-way delay metric, the agent knows that it is required to generate packets. The values for this field are defined in the reference method of measurement (and this frequently results in abbreviated role names such as "Src").

When the Role column of a registry entry defines more than one Role, then the Role SHALL be treated as a Run-time Parameter and supplied for execution. It should be noted that the LMAP framework [RFC7594] distinguishes the Role from other Run-time Parameters, and defines a special parameter "Roles" inside the registry-grouping function list in the LMAP YANG model[RFC8194].

7.4. Output Category

For entries which involve a stream and many singleton measurements, a statistic may be specified in this column to summarize the results to a single value. If the complete set of measured singletons is output, this will be specified here.

Some metrics embed one specific statistic in the reference metric definition, while others allow several output types or statistics.

7.4.1. Type

This column contains the name of the output type. The output type defines a single type of result that the metric produces. It can be the raw results (packet send times and singleton metrics), or it can be a summary statistic. The specification of the output type MUST define the format of the output. In some systems, format specifications will simplify both measurement implementation and collection/storage tasks. Note that if two different statistics are required from a single measurement (for example, both "Xth percentile mean" and "Raw"), then a new output type must be defined ("Xth percentile mean AND Raw"). See the Naming section above for a list of Output Types.

7.4.2. Reference Definition

This column contains a pointer to the specification(s) where the output type and format are defined.

7.4.3. Metric Units

The measured results must be expressed using some standard dimension or units of measure. This column provides the units.

When a sample of singletons (see Section 11 of [RFC2330] for definitions of these terms) is collected, this entry will specify the units for each measured value.

7.4.4. Calibration

Some specifications for Methods of Measurement include the possibility to perform an error calibration. Section 3.7.3 of [RFC7679] is one example. In the registry entry, this field will identify a method of calibration for the metric, and when available, the measurement system SHOULD perform the calibration when requested and produce the output with an indication that it is the result of a calibration method. In-situ calibration could be enabled with an internal loopback that includes as much of the measurement system as possible, performs address manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

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

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

7.5. Administrative information

7.5.1. Status

The status of the specification of this Registered Performance Metric. Allowed values are 'current' and 'deprecated'. All newly defined Information Elements have 'current' status.

7.5.2. Requester

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

7.5.3. Revision

The revision number of a Registered Performance Metric, starting at 0 for Registered Performance Metrics at time of definition and incremented by one for each revision.

7.5.4. Revision Date

The date of acceptance or the most recent revision for the Registered Performance Metric. The date SHALL be determined by IANA and the reviewing Performance Metrics Expert.

7.6. Comments and Remarks

Besides providing additional details which do not appear in other categories, this open Category (single column) allows for unforeseen issues to be addressed by simply updating this informational entry.

8. Processes for Managing the Performance Metric Registry Group

Once a Performance Metric or set of Performance Metrics has been identified for a given application, candidate Performance Metrics Registry entry specifications prepared in accordance with Section 7 should be submitted to IANA to follow the process for review by the Performance Metric Experts, as defined below. This process is also used for other changes to the Performance Metrics Registry, such as deprecation or revision, as described later in this section.

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

8.1. Adding new Performance Metrics to the Performance Metrics Registry

Requests to add Registered Performance Metrics in the Performance Metrics Registry SHALL be submitted to IANA, which forwards the request to a designated group of experts (Performance Metric Experts) appointed by the IESG; these are the reviewers called for by the Specification Required [RFC8126] policy defined for the Performance Metrics Registry. The Performance Metric Experts review the request for such things as compliance with this document, compliance with other applicable Performance Metric-related RFCs, and consistency with the currently defined set of Registered Performance Metrics. The most efficient path for submission begins with preparation of an Internet Draft containing the proposed Performance Metrics Registry entry using the template in Section 11, so that the submission formatting will benefit from the normal IETF Internet Draft submission processing (including HTML-ization).

Submission to IANA may be during IESG review (leading to IETF Standards Action), where an Internet Draft proposes one or more Registered Performance Metrics to be added to the Performance Metrics Registry, including the text of the proposed Registered Performance Metric(s).

If an RFC-to-be includes a Performance Metric and a proposed Performance Metrics Registry entry, but the Performance Metric Expert review determines that one or more of the Section 5 criteria have not been met, then the proposed Performance Metrics Registry entry MUST be removed from the text. Once evidence exists that the Performance Metric meets the criteria in section 5, the proposed Performance Metrics Registry entry SHOULD be submitted to IANA to be evaluated in consultation with the Performance Metric Experts for registration at that time.

Authors of proposed Registered Performance Metrics SHOULD review compliance with the specifications in this document to check their submissions before sending them to IANA.

At least one Performance Metric Expert should endeavor to complete referred reviews in a timely manner. If the request is acceptable, the Performance Metric Experts signify their approval to IANA, and IANA updates the Performance Metrics Registry. If the request is not acceptable, the Performance Metric Experts MAY coordinate with the requester to change the request to be compliant, otherwise IANA SHALL coordinate resolution of issues on behalf of the expert. The Performance Metric Experts MAY choose to reject clearly frivolous or inappropriate change requests outright, but such exceptional circumstances should be rare.

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

Decisions by the Performance Metric Experts may be appealed as in Section 7 of [RFC8126].

8.2. Revising Registered Performance Metrics

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

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

In addition, no policy is defined for revising the Performance Metric entries in the IANA Registry or addressing errors therein. To be clear, changes and deprecations within the Performance Metrics Registry are not encouraged, and should be avoided to the extent possible. However, in recognition that change is inevitable, the provisions of this section address the need for revisions.

Revisions are initiated by sending a candidate Registered Performance Metric definition to IANA, as in Section 8.1, identifying the existing Performance Metrics Registry entry, and explaining how and why the existing entry should be revised.

The primary requirement in the definition of procedures for managing changes to existing Registered Performance Metrics is avoidance of measurement interoperability problems; the Performance Metric Experts must work to maintain interoperability above all else. Changes to Registered Performance Metrics may only be done in an interoperable way; necessary changes that cannot be done in a way to allow interoperability with unchanged implementations MUST result in the creation of a new Registered Performance Metric (with a new Name, replacing the RFCXXXXsecY portion of the name) and possibly the deprecation of the earlier metric.

A change to a Registered Performance Metric SHALL be determined to be backward-compatible when:

1. it involves the correction of an error that is obviously only editorial; or
2. it corrects an ambiguity in the Registered Performance Metric's definition, which itself leads to issues severe enough to prevent the Registered Performance Metric's usage as originally defined; or
3. it corrects missing information in the metric definition without changing its meaning (e.g., the explicit definition of 'quantity' semantics for numeric fields without a Data Type Semantics value); or
4. it harmonizes with an external reference that was itself corrected.

If a Performance Metric revision is deemed permissible and backward-compatible by the Performance Metric Experts, according to the rules in this document, IANA SHOULD execute the change(s) in the Performance Metrics Registry. The requester of the change is appended to the original requester in the Performance Metrics Registry. The Name of the revised Registered Performance Metric, including the RFCXXXsecY portion of the name, SHALL remain unchanged (even when the change is the result of IETF Standards Action; the revised registry entry SHOULD reference the new immutable document, such as an RFC or for other standards bodies, it is likely to be necessary to reference a specific, dated version of a specification, in an appropriate category and column).

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

When a revised Registered Performance Metric is accepted into the Performance Metrics Registry, the date of acceptance of the most recent revision is placed into the revision Date column of the registry for that Registered Performance Metric.

Where applicable, additions to Registered Performance Metrics in the form of text Comments or Remarks should include the date, but such additions may not constitute a revision according to this process.

Older version(s) of the updated metric entries are kept in the registry for archival purposes. The older entries are kept with all fields unmodified (version, revision date) except for the status field that SHALL be changed to "Deprecated".

8.3. Deprecating Registered Performance Metrics

Changes that are not permissible by the above criteria for Registered Performance Metric's revision may only be handled by deprecation. A Registered Performance Metric MAY be deprecated and replaced when:

1. the Registered Performance Metric definition has an error or shortcoming that cannot be permissibly changed as in Section 8.2 Revising Registered Performance Metrics; or
2. the deprecation harmonizes with an external reference that was itself deprecated through that reference's accepted deprecation method.

A request for deprecation is sent to IANA, which passes it to the Performance Metric Experts for review. When deprecating an Performance Metric, the Performance Metric description in the Performance Metrics Registry must be updated to explain the deprecation, as well as to refer to any new Performance Metrics created to replace the deprecated Performance Metric.

The revision number of a Registered Performance Metric is incremented upon deprecation, and the revision Date updated, as with any revision.

The intentional use of deprecated Registered Performance Metrics should result in a log entry or human-readable warning by the respective application.

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

The deprecated entries are kept with all fields unmodified, except the version, revision date, and the status field (changed to "Deprecated").

9. Security considerations

This draft defines a registry structure, and does not itself introduce any new security considerations for the Internet. The definition of Performance Metrics for this registry may introduce some security concerns, but the mandatory references should have their own considerations for security, and such definitions should be reviewed with security in mind if the security considerations are not covered by one or more reference standards.

The aggregated results of the performance metrics described in this registry might reveal network topology information that may be

considered sensitive. If such cases are found, then access control mechanisms should be applied.

10. IANA Considerations

With the background and processes described in earlier sections, this document requests the following IANA Actions.

Editor's Note: Mock-ups of the implementation of this set of requests have been prepared with IANA's help during development of this memo, and have been captured in the Proceedings of IPPM working group sessions. IANA is currently preparing a mock-up. A recent version is available here: <http://encrypted.net/IETFMetricsRegistry-106.html>

10.1. Registry Group

The new registry group SHALL be named, "PERFORMANCE METRICS Group".

Registration Procedure: Specification Required

Reference: <This RFC>

Experts: Performance Metrics Experts

Note: TBD

10.2. Performance Metric Name Elements

This document specifies the procedure for Performance Metrics Name Element Registry setup. IANA is requested to create a new set of registries for Performance Metric Name Elements called "Registered Performance Metric Name Elements". Each Registry, whose names are listed below:

MetricType:

Method:

SubTypeMethod:

Spec:

Units:

Output:

will contain the current set of possibilities for Performance Metrics Registry Entry Names.

To populate the Registered Performance Metric Name Elements at creation, the IANA is asked to use the lists of values for each name element listed in Section 7.1.2. The Name Elements in each registry are case-sensitive.

When preparing a Metric entry for Registration, the developer SHOULD choose Name elements from among the registered elements. However, if the proposed metric is unique in a significant way, it may be necessary to propose a new Name element to properly describe the metric, as described below.

A candidate Metric Entry RFC or immutable document for IANA and Expert Review would propose one or more new element values required to describe the unique entry, and the new name element(s) would be reviewed along with the metric entry. New assignments for Registered Performance Metric Name Elements will be administered by IANA through Specification Required policy (which includes Expert Review) [RFC8126], i.e., review by one of a group of experts, the Performance Metric Experts, who are appointed by the IESG upon recommendation of the Transport Area Directors.

10.3. New Performance Metrics Registry

This document specifies the procedure for Performance Metrics Registry setup. IANA is requested to create a new registry for Performance Metrics called "Performance Metrics Registry". This Registry will contain the following Summary columns:

Identifier:

Name:

URI:

Description:

Reference:

Change Controller:

Version:

Descriptions of these columns and additional information found in the template for registry entries (categories and columns) are further defined in section Section 7.

The Identifier 0 should be Reserved. The Registered Performance Metric unique identifier is an unbounded integer (range 0 to

infinity). The Identifier values from 64512 to 65536 are reserved for private or experimental use, and the user may encounter overlapping uses. When adding newly Registered Performance Metrics to the Performance Metrics Registry, IANA SHOULD assign the lowest available identifier to the new Registered Performance Metric. If a Performance Metrics Expert providing review determines that there is a reason to assign a specific numeric identifier, possibly leaving a temporary gap in the numbering, then the Performance Expert SHALL inform IANA of this decision.

Names starting with the prefix Priv_ are reserved for private use, and are not considered for registration. The "Name" column entries are further defined in section Section 7.

The "URI" column will have a URL to the full template of each registry entry. The Registry Entry text SHALL be HTML-ized to aid the reader, with links to reference RFCs (similar to the way that Internet Drafts are HTML-ized, the same tool can perform the function) or immutable document.

The "Reference" column will include an RFC number, an approved specification designator from another standards body, or other immutable document.

New assignments for Performance Metrics Registry will be administered by IANA through Specification Required policy (which includes Expert Review) [RFC8126], i.e., review by one of a group of experts, the Performance Metric Experts, who are appointed by the IESG upon recommendation of the Transport Area Directors, or by Standards Action. The experts can be initially drawn from the Working Group Chairs, document editors, and members of the Performance Metrics Directorate, among other sources of experts.

Extensions of the Performance Metrics Registry require IETF Standards Action. Only one form of registry extension is envisaged:

1. Adding columns, or both categories and columns, to accommodate unanticipated aspects of new measurements and metric categories.

If the Performance Metrics Registry is extended in this way, the Version number of future entries complying with the extension SHALL be incremented (either in the unit or tenths digit, depending on the degree of extension).

11. Blank Registry Template

This section provides a blank template to help IANA and registry entry writers.

11.1. Summary

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

11.1.1. ID (Identifier)

<insert a numeric identifier, an integer, TBD>

11.1.2. Name

<insert name according to metric naming convention>

11.1.3. URI

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

11.1.4. Description

<provide a description>

11.1.5. Change Controller

11.1.6. Version (of Registry Format)

11.2. Metric Definition

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

11.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

<specific section reference and additional clarifications, if needed>

11.2.2. Fixed Parameters

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

11.3. Method of Measurement

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

11.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

11.3.2. Packet Stream Generation

<list of generation parameters and section/spec references if needed>

11.3.3. Traffic Filtering (observation) Details

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

<section reference>.

11.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

11.3.5. Run-time Parameters and Data Format

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

<list of run-time parameters, and their data formats>

11.3.6. Roles

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

11.4. Output

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

11.4.1. Type

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

11.4.2. Reference Definition

<describe the reference data format for each type of result>

11.4.3. Metric Units

<insert units for the measured results, and the reference specification>.

11.4.4. Calibration

<insert information on calibration>

11.5. Administrative items

11.5.1. Status

<current or deprecated>

11.5.2. Requester

<name or RFC, etc.>

11.5.3. Revision

<1.0>

11.5.4. Revision Date

<format YYYY-MM-DD>

11.6. Comments and Remarks

<Additional (Informational) details for this entry>

12. Acknowledgments

Thanks to Brian Trammell and Bill Cervený, IPPM chairs, for leading some brainstorming sessions on this topic. Thanks to Barbara Stark and Juergen Schoenwaelder for the detailed feedback and suggestions. Thanks to Andrew McGregor for suggestions on metric naming. Thanks to Michelle Cotton for her early IANA review, and to Amanda Barber for answering questions related to the presentation of the registry and accessibility of the complete template via URL. Thanks to Roni

Even for his review and suggestions to generalize the procedures.
Thanks to ~all the Area Directors for their reviews.

13. References

13.1. Normative References

- [RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <<https://www.rfc-editor.org/info/rfc2026>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <<https://www.rfc-editor.org/info/rfc2330>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, DOI 10.17487/RFC6390, October 2011, <<https://www.rfc-editor.org/info/rfc6390>>.
- [RFC6576] Geib, R., Ed., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, DOI 10.17487/RFC6576, March 2012, <<https://www.rfc-editor.org/info/rfc6576>>.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

13.2. Informative References

- [I-D.ietf-ippm-initial-registry]
Morton, A., Bagnulo, M., Eardley, P., and K. D'Souza,
"Initial Performance Metrics Registry Entries", draft-
ietf-ippm-initial-registry-15 (work in progress), December
2019.
- [RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip
Delay Metric for IPPM", RFC 2681, DOI 10.17487/RFC2681,
September 1999, <<https://www.rfc-editor.org/info/rfc2681>>.
- [RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network
performance measurement with periodic streams", RFC 3432,
DOI 10.17487/RFC3432, November 2002,
<<https://www.rfc-editor.org/info/rfc3432>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
Jacobson, "RTP: A Transport Protocol for Real-Time
Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550,
July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed.,
"RTP Control Protocol Extended Reports (RTCP XR)",
RFC 3611, DOI 10.17487/RFC3611, November 2003,
<<https://www.rfc-editor.org/info/rfc3611>>.
- [RFC4148] Stephan, E., "IP Performance Metrics (IPPM) Metrics
Registry", BCP 108, RFC 4148, DOI 10.17487/RFC4148, August
2005, <<https://www.rfc-editor.org/info/rfc4148>>.
- [RFC5474] Duffield, N., Ed., Chiou, D., Claise, B., Greenberg, A.,
Grossglauser, M., and J. Rexford, "A Framework for Packet
Selection and Reporting", RFC 5474, DOI 10.17487/RFC5474,
March 2009, <<https://www.rfc-editor.org/info/rfc5474>>.
- [RFC5475] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F.
Raspall, "Sampling and Filtering Techniques for IP Packet
Selection", RFC 5475, DOI 10.17487/RFC5475, March 2009,
<<https://www.rfc-editor.org/info/rfc5475>>.
- [RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G.
Carle, "Information Model for Packet Sampling Exports",
RFC 5477, DOI 10.17487/RFC5477, March 2009,
<<https://www.rfc-editor.org/info/rfc5477>>.

- [RFC6035] Pendleton, A., Clark, A., Johnston, A., and H. Sinnreich, "Session Initiation Protocol Event Package for Voice Quality Reporting", RFC 6035, DOI 10.17487/RFC6035, November 2010, <<https://www.rfc-editor.org/info/rfc6035>>.
- [RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, DOI 10.17487/RFC6248, April 2011, <<https://www.rfc-editor.org/info/rfc6248>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <<https://www.rfc-editor.org/info/rfc7012>>.
- [RFC7014] D'Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow Selection Techniques", RFC 7014, DOI 10.17487/RFC7014, September 2013, <<https://www.rfc-editor.org/info/rfc7014>>.
- [RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <<https://www.rfc-editor.org/info/rfc7594>>.
- [RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <<https://www.rfc-editor.org/info/rfc7679>>.
- [RFC8194] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", RFC 8194, DOI 10.17487/RFC8194, August 2017, <<https://www.rfc-editor.org/info/rfc8194>>.

Authors' Addresses

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

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

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Email: bclaise@cisco.com

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich
ENGLAND

Email: philip.eardley@bt.com

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

Email: acmorton@att.com

Aamer Akhter
Consultant
118 Timber Hitch
Cary, NC
USA

Email: aakhter@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 12, 2016

G. Mirsky
Ericsson
I. Meilik
Broadcom
February 9, 2016

Support of IEEE-1588 time stamp format in Two-Way Active Measurement
Protocol (TWAMP)
draft-mirsky-ippm-time-format-03

Abstract

This document describes an OPTIONAL feature for active performance measurement protocols allowing use of time stamp format defined in IEEE-1588v2-2008.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 12, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Conventions used in this document	3
1.1.1. Terminology	3
1.1.2. Requirements Language	3
2. OWAMP and TWAMP Extensions	3
2.1. Timestamp Format Negotiation in Setting Up Connection in OWAMP	4
2.2. Timestamp Format Negotiation in Setting Up Connection in TWAMP	5
2.3. OWAMP-Test and TWAMP-Test Update	5
2.3.1. Consideration for TWAMP Light mode	6
3. IANA Considerations	6
4. Security Considerations	6
5. Acknowledgements	6
6. Normative References	7
Authors' Addresses	7

1. Introduction

One-Way Active Measurement Protocol (OWAMP) [RFC4656] defines that only the NTP [RFC5905] format of a time stamp can be used in OWAMP-Test protocol. Two-Way Active Measurement Protocol (TWAMP) [RFC5357] adopted the OWAMP-Test packet format and extended it by adding a format for a reflected test packet. Both the sender's and reflector's packets time stamps are expected to follow the 64-bit long NTP format [RFC5905]. NTP, when used over Internet, typically achieves clock accuracy of about 5ms to 100ms. Surveys conducted recently suggest that 90% devices achieve accuracy of better than 100 ms and 99% - better than 1 sec. It should be noted that NTP synchronizes clocks on the control plane, not on data plane. Distribution of clock within a node may be supported by independent NTP domain or via interprocess communication in multiprocessor distributed system. And of mentioned solutions will be subject to additional queuing delays that negatively affect data plane clock accuracy.

Precision Time Protocol (PTP) [IEEE.1588.2008] has gained wide support since the development of OWAMP and TWAMP. PTP, using on-path support and other mechanisms, allows sub-microsecond clock accuracy. PTP is now supported in multiple implementations of fast forwarding engines and thus accuracy achieved by PTP is the accuracy of clock in data plane. Thus providing option to use more accurate clock as source of time stamps for IP performance measurement is one of advantages this proposal helps to achieve. Another advantage realized by simplification of hardware in data plane. To support OWAMP or TWAMP test protocol time stamps must be converted from PTP to NTP.

That requires resources, use of micro-code or additional processing elements, that are always limited. To address this, this document proposes optional extensions to Control and Test protocols to support use of IEEE-1588v2 time stamp format as optional alternative to the NTP time stamp format.

One of the goals of this proposal is not only allow end-points of a test session to use other than NTP timestamp but to support backwards compatibility with nodes that do not yet support this extension.

1.1. Conventions used in this document

1.1.1. Terminology

IPPM: IP Performance Measurement

NTP: Network Time Protocol

PTP: Precision Time Protocol

TWAMP: Two-Way Active Measurement Protocol

OWAMP: One-Way Active Measurement Protocol

1.1.2. Requirements Language

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

2. OWAMP and TWAMP Extensions

OWAMP connection establishment follows the procedure defined in Section 3.1 of [RFC4656] and additional steps in TWAMP described in Section 3.1 of [RFC5357]. In these procedures the Modes field been used to identify and select specific communication capabilities. At the same time the Modes field been recognized and used as extension mechanism [RFC6038]. The new feature requires one bit position for Server and Control-Client to negotiate which timestamp format can be used in some or all test sessions invoked with this control connection. The end-point of the test session, Session-Sender and Session-Receiver or Session-Reflector, that supports this extension MUST be capable to interpret NTP and PTPv2 timestamp formats. If the end-point does not support this extension, then the value of PTPv2 Timestamp flag MUST be 0 because it is in Must Be Zero field. If value of PTPv2 Timestamp flags is 0, then the advertising node can use and interpret only NTP timestamp format.

Use of PTPv2 Timestamp flags discussed in the following sub-sections. For details on the assigned values and bit positions see the Section 3.

2.1. Timestamp Format Negotiation in Setting Up Connection in OWAMP

In OWAMP-Test [RFC4656] it is the Session-Receiver and/or Fetch-Client that are interpreting collected timestamps. Thus announced by a Server in the Modes field timestamp format indicates which formats the Session-Receiver is capable to interpret. The Control-Client inspects values set by the Server for timestamp formats and sets values in the Modes field of the Set-Up-Response message according to timestamp formats Session-Sender is capable of using. The rules of setting timestamp flags in Modes field in server greeting and Set-Up-Response messages and interpreting them are as follows:

- o The Server that establishes test sessions for Session-Receiver that supports this extension MUST set PTPv2 Timestamp flag to 1 in the server greeting message according to the requirement listed in Section 2.
- o If PTPv2 Timestamp flag of the server greeting message that the Control-Client receives has value 0, then the Session-Sender MUST use NTP format for timestamp in the test session and Control-Client SHOULD set PTPv2 Timestamp flag to 0 in accordance with [RFC4656]. If the Session-Sender cannot use NTP timestamps, then the Control-Client SHOULD close the TCP connection associated with the OWAMP-Control session.
- o If the Session-Sender can set timestamp in PTPv2 format, then the Control-Client MUST set the PTPv2 Timestamp flag to 1 in Modes field in the Set-Up-Response message and the Session-Sender MUST set timestamp in PTPv2 timestamp format. Otherwise the Control-Client MUST set the PTPv2 Timestamp flag in the Set-Up-Response message to 0.
- o Otherwise, if the Session-Sender can set timestamp in NTP format, then the Session-Sender MUST set timestamp in NTP timestamp format. Otherwise the Control-Client SHOULD close the TCP connection associated with the OWAMP-Control session..

If values of both NTP and PTPv2 Timestamp flags in the Set-Up-Response message are equal to 0, then that indicates that the Control-Client can set timestamp only in NTP format.

If OWAMP-Control uses Fetch-Session commands, then selection and use of one or another timestamp format is local decision for both Session-Sender and Session-Receiver.

2.2. Timestamp Format Negotiation in Setting Up Connection in TWAMP

In TWAMP-Test [RFC5357] it is the Session-Sender that is interpreting collected timestamps. Hence, in the Modes field a Server advertises timestamp formats that the Session-Reflector can use in TWAMP-Test message. The choice of the timestamp format to be used by the Session-Sender is a local decision. The Control-Client inspects the Modes field and sets timestamp flags values to indicate which format will be used by the Session-Reflector. The rules of setting and interpreting flag values are as follows:

- o Server MUST set to 1 value of PTPv2 Timestamp flag in its greeting message if Session-Reflector can set timestamp in PTPv2 format. Otherwise the PTPv2 Timestamp flag MUST be set to 0.
- o If value of the PTPv2 Timestamp flag in received server greeting message equals 0, then Session-Reflector does not support this extension and will use NTP timestamp format. Control-Client SHOULD set PTPv2 Timestamp flag to 0 in Set-Up-Response message in accordance with [RFC5357].
- o Control-Client MUST set PTPv2 Timestamp flag value to 1 in Modes field in the Set-Up-Response message if Server advertised ability of the Session-Reflector to use PTPv2 format for timestamps. Otherwise the flag MUST be set to 0.
- o If the values of PTPv2 Timestamp flag in the Set-Up-Response message equals 0, then that means that Session-Sender can only interpret NTP timestamp format. Then the Session-Reflector MUST use NTP timestamp format. If the Session-Reflector does not support NTP format for timestamps then Server and SHOULD close the TCP connection associated with the TWAMP-Control session.

2.3. OWAMP-Test and TWAMP-Test Update

Participants of a test session need to indicate which timestamp format being used. The proposal is to use Z field in Error Estimate defined in Section 4.1.2 of [RFC4656]. The new interpretation of the Error Estimate is in addition to it specifying error estimate and synchronization, Error Estimate indicates format of a collected timestamp. And this proposal changes the semantics of the Z bit field, the one between S and Scale fields, to be referred as Timestamp format and value MUST be set according to the following:

- o 0 - NTP 64 bit format of a timestamp;
- o 1 - PTPv2 truncated format of a timestamp.

As result of this value of the Z field from Error Estimate, Sender Error Estimate or Send Error Estimate and Receive Error Estimate SHOULD NOT be ignored and MUST be used when calculating delay and delay variation metrics based on collected timestamps.

2.3.1. Consideration for TWAMP Light mode

This document does not specify how Session-Sender and Session-Reflector in TWAMP Light mode are informed of timestamp format to be used. It is assumed that, for example, configuration could be used to direct Session-Sender and Session-Reflector respectively to use timestamp format according to their capabilities and rules listed in Section 2.2.

3. IANA Considerations

The TWAMP-Modes registry defined in [RFC5618].

IANA is requested to reserve a new PTPv2 Timestamp as follows:

Value	Description	Semantics	Reference
TBA1 (proposed 256)	PTPv2 Timestamp Capability	bit position TBA2 (proposed 8)	This document

Table 1: New Timestamp Capability

4. Security Considerations

Use of particular format of a timestamp in test session does not appear to introduce any additional security threat to hosts that communicate with OWAMP and/or TWAMP as defined in [RFC4656], [RFC5357] respectively. The security considerations that apply to any active measurement of live networks are relevant here as well. See the Security Considerations sections in [RFC4656] and [RFC5357].

5. Acknowledgements

The authors would like to thank Lakshmikanthan and Suchit Bansal for their insightful suggestions. The authors would like to thank David Allan for his thorough review and thoughtful comments.

6. Normative References

- [IEEE.1588.2008]
"Standard for a Precision Clock Synchronization Protocol
for Networked Measurement and Control Systems",
IEEE Standard 1588, March 2008.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M.
Zekauskas, "A One-way Active Measurement Protocol
(OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006,
<<http://www.rfc-editor.org/info/rfc4656>>.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J.
Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)",
RFC 5357, DOI 10.17487/RFC5357, October 2008,
<<http://www.rfc-editor.org/info/rfc5357>>.
- [RFC5618] Morton, A. and K. Hedayat, "Mixed Security Mode for the
Two-Way Active Measurement Protocol (TWAMP)", RFC 5618,
DOI 10.17487/RFC5618, August 2009,
<<http://www.rfc-editor.org/info/rfc5618>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch,
"Network Time Protocol Version 4: Protocol and Algorithms
Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010,
<<http://www.rfc-editor.org/info/rfc5905>>.
- [RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement
Protocol (TWAMP) Reflect Octets and Symmetrical Size
Features", RFC 6038, DOI 10.17487/RFC6038, October 2010,
<<http://www.rfc-editor.org/info/rfc6038>>.

Authors' Addresses

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Israel Meilik
Broadcom

Email: israel@broadcom.com

Network Working Group
Internet-Draft
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.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Scope	3
3. Packets of Type-P	3
4. Standard-Formed Packets	5
5. Conclusions	7
6. Security Considerations	7
7. IANA Considerations	7
8. Acknowledgements	8
9. References	8
9.1. Normative References	8
9.2. Informative References	10
Authors' Addresses	10

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.

- o 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).
- o 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:

- o Extension Header inspection: Some intermediate nodes may inspect Extension Headers or the entire IPv6 packet while in transit. In exceptional cases, they may drop the packet or route via a sub-optimal path, and measurements may be unreliable or unrepeatable. The packet (if it arrives) may be standard-formed, with a corresponding Type-P.
- o Extension Header modification: In Hop-by-Hop headers, some TLV encoded options may be permitted to change at intermediate nodes while in transit. The resulting packet may be standard-formed, with a corresponding Type-P.
- o Extension Header insertion or deletion: 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.
- o A change in packet length (from the corresponding packet observed at the Source) or header modification is a significant factor in Internet measurement, and requires a new Type-P to be reported.

We further require that if a packet is described as having a "length of B octets", then $0 \leq B \leq 65535$; and if B is the payload length in octets, then $B \leq (65535 - \text{IP header size in octets, including any Extension Headers})$. 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

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 24, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	7
2. Scope	7
3. Registry Categories and Columns	8
4. UDP Round-trip Latency Registry Entry	8
4.1. Summary	9
4.1.1. ID (Identifier)	9
4.1.2. Name	9
4.1.3. URIs	9
4.1.4. Description	9
4.2. Metric Definition	9
4.2.1. Reference Definition	9
4.2.2. Fixed Parameters	10
4.3. Method of Measurement	11
4.3.1. Reference Method	11
4.3.2. Packet Generation Stream	12
4.3.3. Traffic Filtering (observation) Details	13
4.3.4. Sampling Distribution	13
4.3.5. Run-time Parameters and Data Format	13
4.3.6. Roles	14
4.4. Output	14
4.4.1. Type	14
4.4.2. Data Format	15
4.4.3. Reference	15
4.4.4. Metric Units	15
4.5. Administrative items	15
4.5.1. Status	15
4.5.2. Requestor (keep?)	16
4.5.3. Revision	16
4.5.4. Revision Date	16
4.6. Comments and Remarks	16
5. Packet Delay Variation Registry Entry	16

5.1.	Summary	16
5.1.1.	ID (Identifier)	16
5.1.2.	Name	16
5.1.3.	URI	17
5.1.4.	Description	17
5.2.	Metric Definition	17
5.2.1.	Reference Definition	17
5.2.2.	Fixed Parameters	17
5.3.	Method of Measurement	18
5.3.1.	Reference Method	18
5.3.2.	Packet Generation Stream	18
5.3.3.	Traffic Filtering (observation) Details	18
5.3.4.	Sampling Distribution	19
5.3.5.	Run-time Parameters and Data Format	19
5.3.6.	Roles	19
5.4.	Output	19
5.4.1.	Type/Value (two diff terms used)	19
5.4.2.	Data Format	20
5.4.3.	Reference	21
5.4.4.	Metric Units	21
5.5.	Administrative items	21
5.5.1.	Status	21
5.5.2.	Requestor (keep?)	21
5.5.3.	Revision	21
5.5.4.	Revision Date	21
5.6.	Comments and Remarks	22
6.	DNS Response Latency Registry Entry	22
6.1.	Summary	22
6.1.1.	ID (Identifier)	22
6.1.2.	Name	22
6.1.3.	URI	22
6.1.4.	Description	22
6.2.	Metric Definition	22
6.2.1.	Reference Definition	23
6.2.2.	Fixed Parameters	23
6.3.	Method of Measurement	25
6.3.1.	Reference Method	25
6.3.2.	Packet Generation Stream	26
6.3.3.	Traffic Filtering (observation) Details	26
6.3.4.	Sampling Distribution	26
6.3.5.	Run-time Parameters and Data Format	26
6.3.6.	Roles	27
6.4.	Output	27
6.4.1.	Type/Value (two diff terms used)	28
6.4.2.	Data Format	28
6.4.3.	Reference	29
6.4.4.	Metric Units	29
6.5.	Administrative items	29

6.5.1.	Status	29
6.5.2.	Requestor (keep?)	29
6.5.3.	Revision	29
6.5.4.	Revision Date	29
6.6.	Comments and Remarks	29
7.	UDP Poisson One-way Delay Registry Entries	30
7.1.	Summary	30
7.1.1.	ID (Identifier)	30
7.1.2.	Name	30
7.1.3.	URI and URL	30
7.1.4.	Description	31
7.2.	Metric Definition	31
7.2.1.	Reference Definition	31
7.2.2.	Fixed Parameters	31
7.3.	Method of Measurement	32
7.3.1.	Reference Method	32
7.3.2.	Packet Generation Stream	32
7.3.3.	Traffic Filtering (observation) Details	33
7.3.4.	Sampling Distribution	33
7.3.5.	Run-time Parameters and Data Format	33
7.3.6.	Roles	34
7.4.	Output	34
7.4.1.	Type/Value (two diff terms used)	34
7.4.2.	Data Format	34
7.4.3.	Reference	36
7.4.4.	Metric Units	36
7.5.	Administrative items	37
7.5.1.	Status	37
7.5.2.	Requestor (keep?)	37
7.5.3.	Revision	37
7.5.4.	Revision Date	37
7.6.	Comments and Remarks	37
8.	UDP Periodic One-way Delay Registry Entries	37
8.1.	Summary	37
8.1.1.	ID (Identifier)	37
8.1.2.	Name	38
8.1.3.	URI and URL	38
8.1.4.	Description	38
8.2.	Metric Definition	38
8.2.1.	Reference Definition	38
8.2.2.	Fixed Parameters	39
8.3.	Method of Measurement	40
8.3.1.	Reference Method	40
8.3.2.	Packet Generation Stream	40
8.3.3.	Traffic Filtering (observation) Details	41
8.3.4.	Sampling Distribution	41
8.3.5.	Run-time Parameters and Data Format	41
8.3.6.	Roles	42

8.4.	Output	42
8.4.1.	Type/Value (two diff terms used)	42
8.4.2.	Data Format	42
8.4.3.	Reference	44
8.4.4.	Metric Units	44
8.5.	Administrative items	44
8.5.1.	Status	44
8.5.2.	Requestor (keep?)	44
8.5.3.	Revision	44
8.5.4.	Revision Date	45
8.6.	Comments and Remarks	45
9.	partly BLANK Registry Entry	45
9.1.	Summary	45
9.1.1.	ID (Identifier)	45
9.1.2.	Name	45
9.1.3.	URI	45
9.1.4.	Description	45
9.2.	Metric Definition	45
9.2.1.	Reference Definition	45
9.2.2.	Fixed Parameters	46
9.3.	Method of Measurement	47
9.3.1.	Reference Method	47
9.3.2.	Packet Generation Stream	47
9.3.3.	Traffic Filtering (observation) Details	47
9.3.4.	Sampling Distribution	47
9.3.5.	Run-time Parameters and Data Format	47
9.3.6.	Roles	48
9.4.	Output	48
9.4.1.	Type/Value (two diff terms used)	48
9.4.2.	Data Format	48
9.4.3.	Reference	48
9.4.4.	Metric Units	48
9.5.	Administrative items	48
9.5.1.	Status	48
9.5.2.	Requestor (keep?)	48
9.5.3.	Revision	49
9.5.4.	Revision Date	49
9.6.	Comments and Remarks	49
10.	BLANK Registry Entry	49
10.1.	Summary	49
10.1.1.	ID (Identifier)	49
10.1.2.	Name	49
10.1.3.	URI	49
10.1.4.	Description	49
10.2.	Metric Definition	49
10.2.1.	Reference Definition	50
10.2.2.	Fixed Parameters	50
10.3.	Method of Measurement	50

10.3.1.	Reference Method	50
10.3.2.	Packet Generation Stream	50
10.3.3.	Traffic Filtering (observation) Details	50
10.3.4.	Sampling Distribution	50
10.3.5.	Run-time Parameters and Data Format	50
10.3.6.	Roles	50
10.4.	Output	51
10.4.1.	Type/Value (two diff terms used)	51
10.4.2.	Data Format	51
10.4.3.	Reference	51
10.4.4.	Metric Units	51
10.5.	Administrative items	51
10.5.1.	Status	51
10.5.2.	Requestor (keep?)	51
10.5.3.	Revision	51
10.5.4.	Revision Date	51
10.6.	Comments and Remarks	51
11.	Example RTCP-XR Registry Entry	52
11.1.	Registry Indexes	52
11.1.1.	Identifier	52
11.1.2.	Name	52
11.1.3.	URI	52
11.1.4.	Status	52
11.1.5.	Requestor	52
11.1.6.	Revision	52
11.1.7.	Revision Date	52
11.1.8.	Description	52
11.1.9.	Reference Specification(s)	53
11.2.	Metric Definition	53
11.2.1.	Reference Definition	53
11.2.2.	Fixed Parameters	53
11.3.	Method of Measurement	54
11.3.1.	Reference Method	54
11.3.2.	Stream Type and Stream Parameters	54
11.3.3.	Output Type and Data Format	54
11.3.4.	Metric Units	54
11.3.5.	Run-time Parameters and Data Format	55
11.4.	Comments and Remarks	56
12.	Revision History	56
13.	Security Considerations	57
14.	IANA Considerations	57
15.	Acknowledgements	57
16.	References	57
16.1.	Normative References	58
16.2.	Informative References	59
	Authors' Addresses	61

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 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 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:

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

- o UDP Payload
 - * total of 9 bytes

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 = 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), $F(\text{Percentile95}) \geq 95\%$ 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>

- o 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.

- o L, a packet length in bits. L = 200 bits.
- o 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.
- o 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.

- o lambda, a rate in reciprocal seconds (for Poisson Streams).
lambda = 1 packet per second
- o 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)>.

- o Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)
- o Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)
- o 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.
- o 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

- o T, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])
- o Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

Raw -

- o 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]).
- o 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]).
- o 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]).
- o 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 deprecated>

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:

- o IPv4 header values:

- * DSCP: set to 0

- * TTL set to 255

- * Protocol: Set to 17 (UDP)
- o UDP header values:
 - * Source port: 53
 - * Destination port: 53
 - * Checksum: the checksum must be calculated
- o 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 disambiguate 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 proceeding.

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 described 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 λ 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>

- o Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- o Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)
- o 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.
- o 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.
- o 1/lambda, average packet rate (for Poisson Streams). (1/lambda = 0.1 packet per second, if fixed)
- o Upper limit on Poisson distribution (values above this limit will be clipped and set to the limit value). (if fixed, Upper limit = 300 seconds.)
- o 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:

- o T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])
- o 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:

- o T, the time when the packet was sent from Src, 128-bit NTP Date Format, see section 6 of [RFC5905])
- o 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.
- o dT is undefined when the packet is not received at Src in waiting time Tmxax seconds (need undefined code for no-response or unsuccessful 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 deprecated>

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:

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

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 disambiguate 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 described 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>

- o Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)
- o Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)
- o 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.
- o 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.
- o 1/lambda, average packet rate (for Poisson Streams). (1/lambda = 1 packet per second, if fixed)

- o 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 ---

- o T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])
- o 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:

$$\text{Max} = (\text{FiniteDelay} [j])$$

such that for some index, j , where $1 \leq j \leq N$
 $\text{FiniteDelay}[j] \geq \text{FiniteDelay}[n]$ 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 deprecated>

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:

- o IPv4 header values:

- * DSCP: set to 0
- * TTL set to 255
- * Protocol: Set to 17 (UDP)

- o UDP header values:

- * Checksum: the checksum must be calculated

- 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

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 disambiguate 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 described 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.

- o incT, the nominal duration of inter-packet interval, first bit to first bit
- o dT, the duration of the interval for allowed sample start times
- o 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>

- o Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)
- o Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)
- o 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.
- o 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.
- o incT, the nominal duration of inter-packet interval, first bit to first bit
- o 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 ---

- o T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])
- o 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:

$$\text{Max} = (\text{FiniteDelay}[j])$$

such that for some index, j , where $1 \leq j \leq N$
 $\text{FiniteDelay}[j] \geq \text{FiniteDelay}[n]$ 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 deprecated>

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:

- o IPv4 header values:

- * DSCP: set to 0
- * TTL set to 255
- * Protocol: Set to 17 (UDP)

- o UDP header values:

- * Checksum: the checksum must be calculated

- o 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 described 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>

- o Value:
- o Data Format: (There may be some precedent to follow here, but otherwise use 64-bit NTP Timestamp Format, see section 6 of [RFC5905]).
- o 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 deprecated>

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 deprecated>

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.

11.1.1.9. 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 0xFFFFF 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.

- o Value: Packets Discarded in Bursts
- o Data Format: 24 bits
- o 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 updates.

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 of "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 (SDS) 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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 18, 2017

Srivathsa. Sarangapani
Peyush. Gupta
Juniper Networks
V. Hegde
Consultant
Q. Wu
Huawei
July 17, 2016

KPI Metrics for Service Monitoring using TWAMP
draft-spv-ippm-monitor-implementation-services-kpi-02

Abstract

We are using a new method to calculate services KPIs and metrics in the network using TWAMP protocol. This draft outlines the implementation of the service KPIs and there use cases in the service plane in the network. The KPIs discussed in this draft include Service Latency and Application Liveliness detection.

Service latency is defined as the time spent by the packet when it is injected in the service module or service card till the time, serviced packet is received back by the TWAMP server. TWAMP server records the timestamp of the packet when it is injected into the service module and then again record the timestamp when it receives the packet after service is applied in the data plane.

Application Liveliness detection means whether the application is up and running in the network. In case you want to monitor the http application or the dns server and verify if they are up and running, this method is applicable. The implementation can be used for liveliness detection of any service in the network.

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 January 18, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Conventions used in this document	3
1.1.1. Requirements Language	3
1.2. Terminology	3
2. Services KPIs	4
2.1. Services Keepalive Monitoring	4
2.2. Service Latency	5
3. Acknowledgements	7
4. IANA Considerations	7
5. Security Considerations	8
6. Normative References	8
Authors' Addresses	9

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.

TWAMP server is used to inject the packet in the service plane for calculating the latency and liveliness. This is done as part of TWAMP data connection. The packets being sent out of TWAMP server is a UDP packet carrying the payload for the service for which we are interested in calculating the KPIs. The timestamping is done at the

TWAMP server. Based on the service model, TWAMP server may be running on the same box where the service is hosted or in a remote server.

The Interface between the TWAMP server and the service plane is implementation specific. the underlying transport is UDP since this is in data path. In this draft, the use cases presented are service latency and keep alive monitoring. Service latency for services like DPI, TDF, Video Caching is calculated. Similarly liveliness for http server, dns application is calculated in the implementation part.

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

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

PDU: Protocol Data Unit

2. Services KPIs

2.1. Services Keepalive Monitoring

Metric Name: Services Keepalive Monitoring

Metric Description: This indicates whether the service is running or not at any point of time.

Method of Measurement or Calculation: The Session-Reflector SHALL inject the Service PDU to the Service Block for service processing. Based on whether the Session-Reflector received the response, the Session-Reflector SHALL decide whether the Service is alive or not.

Units of Measurement: This metric is expressed as a single bit boolean value. If this bit is set then it indicates that the Service Block is functional. If this bit is NOT set then it indicates that the Service Block is not functional.

Measurement Point(s) with Potential Measurement Domain: This metric is calculated at the Session-Reflector.

Measurement Timing: This metric is an instantaneous value. Based on the kind of service it MAY be a good idea to store the history of this value. It can be stored as an average of last one hour for 24 hours, then average of all values over previous day, week, month, year etc. These data MAY be used for some analytics.

Implementation: The Session-Sender SHALL send the Service PDU as part of the TWAMP-Test Packet Padding. When Session-Reflector receives the TWAMP-Test packet, it SHALL extract the Service PDU. The Session-Reflector SHALL extract the service PDU from the TWAMP Payload and inject it to the service module. For ex, incase of http server, the service PDU can just be a http req. The service module will apply services on this PDU and once service processing is done, it would send the response(http resp) back to Session Reflector. The Session Reflector SHALL now reply back to the TWAMP client/session sender with the actual TWAMP data packet with payload being the boolean flag and response service PDU(http resp).

Verification: The metric value is a boolean which SHOULD be either 0(Service NOT Alive) or 1(Service is Alive).

The Session-Reflector MUST start the Packet Padding with the below 4 octets as indicated below. This is followed by the Service PDU (which MAY be same as whatever was sent by Session-Sender or can be the reply/response packet of the Service Block).

Setting Bit 0(X) indicates that the Session-Reflector successfully sent the Service Request to the Service Block and received the response from the Service Block. If this bit is NOT set then it indicates that the Service Block is not functional.

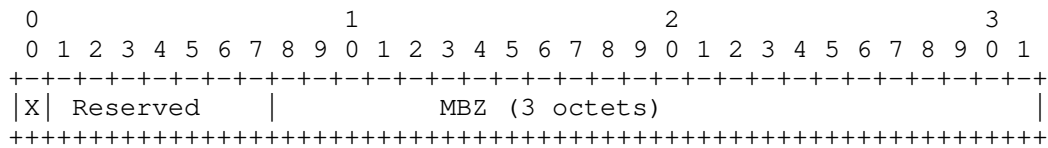


Figure 1: Services Keepalive Monitoring

Use and Applications: This metric is useful for monitoring the liveliness of a service. Normally the liveliness of the Server or a Network Element is not enough to know whether the Service is really Alive or NOT. Say for example if there is a Web Server Application, then just monitoring whether the Server (where the web server) is alive or not by using ping is just not enough to say whether the application is really Alive or not. There could be instances where in the Server is up and running, while the Web Server Application is not running because of some software bug. These kinds of scenarios can be caught only by probing the application and not just the Server where Application is running.

Reporting Model: This metric needs to be associated with a defined time interval, which could be defined by fixed intervals. Based on need, the TWAMP client can negotiate to check the liveliness of a service during connection establishment. If so, then for each of the data packet, the liveliness of the service is measured and reported back to session-sender/client for the entire session.

2.2. Service Latency

Metric Name: Service Latency

Metric Description: This indicates the total latency introduced by the service for a data packet which is undergoing specific treatment offered by that service node in the path. Please note that the latency calculation is service agnostic. Service Latency SHALL include the transit time and actual service time. The transit time should refer to round trip time on the path between Session-Reflector

and service node. The service time should refer to service processing time or service treatment time.

Method of Measurement or Calculation: The Session-Reflector SHALL notedown the time: Service latency measurement Sender Timestamp and inject the Service PDU to the Service Block for service processing. When the Session-Reflector receives the response, the Session-Reflector SHALL notedown the time: Service latency measurement Receiver Timestamp.

Units of Measurement: This metric is expressed as a timestamp which is 64 bit value. The format of the timestamp is the same as in [RFC1305] 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.

So, Timestamp is represented as follows:

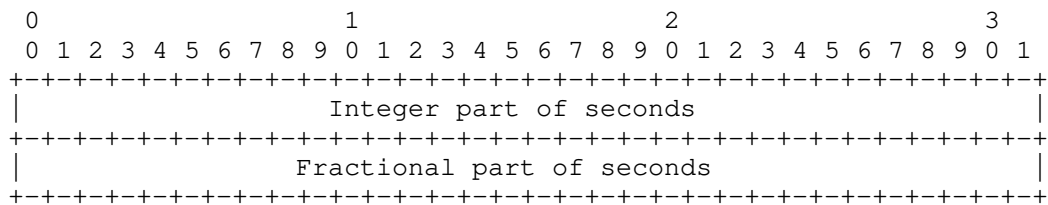


Figure 2: Timestamp Format

Measurement Point(s) with Potential Measurement Domain: This metric is calculated at the Session-Reflector.

Measurement Timing: This metric is an instantaneous value. Based on the kind of service it MAY be a good idea to store the history of this value. It can be stored as an average of last one hour for 24 hours, then average of all values over previous day, week, month, year etc. These data MAY be used for some analytics.

Implementation: The Session-Reflector SHALL extract the service PDU from the TWAMP Payload and inject it to the service module. For ex, incase of http server, the service PDU can just be a http req. The injecting of service module can be broken into below steps:

- Session Reflector should note down the time(say T5) at which this service PDU(http req) is injected to service module.

- The service module will apply services on this PDU and once service processing is done, it would send the response(http resp) back to Session Reflector.
- Once this response serviced PDU is received at Session Reflector, the time(say T6) SHOULD be noted.
- The Session Reflector SHALL now reply back to the TWAMP client/session sender with the actual TWAMP data packet with payload being the 2 timestamps and the response service PDU(http resp).
- Now the TWAMP client/session sender after receiving the packet back can calculate the service latency, which would be T6-T5.

Verification: This metric value is a pair of timestamps. Service Latency will be calculated by subtracting the "Service latency measurement Sender Timestamp" from the "Service latency measurement Receiver Timestamp"

The Session-Reflector MUST start the Packet Padding with the 16 octets indicated below. This SHALL be followed by the Service PDU(which MAY be same as whatever was sent by Session-Sender or can be the reply/response packet of the Service Block).

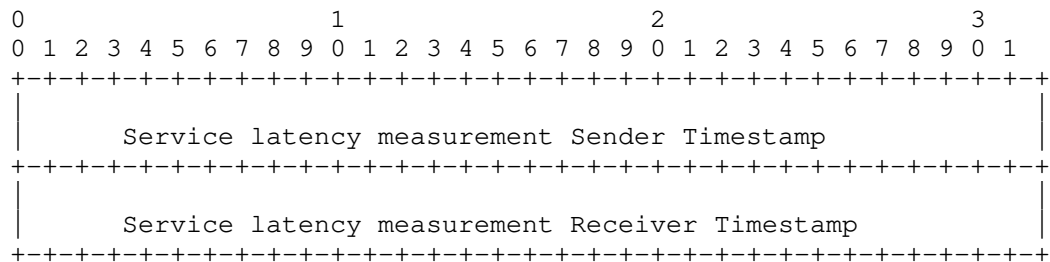


Figure 3: Services Keepalive Monitoring

3. Acknowledgements

We would like to thank Perceival A Monteiro for their comments, suggestions, reviews, helpful discussion, and proof-reading

4. IANA Considerations

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

Table 1: 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 2: New Services KPIs Monitoring Capability

5. Security Considerations

The TWAMP protocol (RFC 5357) supports authenticated and encrypted mode for TWAMP session and data. The implementation discussed in the proposed extension supports the authenticated and encrypted mode and is therefore provides a secure mechanism to monitor services KPIs in the network.

6. 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>>.
- [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>>.
- [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
89, Asthagrama Layout 2nd Stage, Basavehwaranagar
Bangalore 560079
INDIA

Phone: +91 9845052354
Email: srivathsas@juniper.net

Peyush Gupta
Juniper Networks
Flat #206, 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
URI: <http://www.vinayakhegde.com>

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Phone: +86-25-84565892
Email: bill.wu@huawei.com

IP Performance Metrics Group
Internet-Draft
Intended status: Standards Track
Expires: January 18, 2017

Srivathsa. Sarangapani
Peyush. Gupta
Juniper Networks
V. Hegde
Consultant
Q. Wu
Huawei
July 17, 2016

Monitoring Service KPIs using TWAMP - Methodology
draft-spv-ippm-monitor-methodology-services-kpi-02

Abstract

The TWAMP protocol provides a common architecture for two way measurements in the IP network. However IP network performance are also affected by a set of L4-L7 service deployed in the network. Monitoring of these service performance in the IP network also plays a vital role in network optimization and application layer traffic optimization. This capability is not supported by the existing TWAMP protocol.

In this document, we extend TWAMP protocol to support service performance monitoring and service KPIs calculation. Some of the existing fields in the TWAMP protocol are extended to support new modes for calculating these KPIs. A set of new messages are added in the control protocol between TWAMP client (session sender) and the TWAMP server (session reflector). Services here ranging from Layer 4 to Layer 7 services, such as Http based services, Traffic load balancer, DPI, Video caching, real time streaming and IPSec. The KPIs MAY be service latency, liveliness of an application, number of flows and sessions per service, load balancer statistics.

There is a separate Draft[I.D-spv-ippm-monitor-implementation-services-kpi] that talks about implementation of monitoring these KPIs in the network using TWAMP. Monitoring of these KPIs in the service plane with in a network play a vital role in optimum usage of network resources and improving the overall performance and capacity.

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 January 18, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Conventions used in this document	3
1.1.1. Requirements Language	3
1.2. Terminology	3
2. Purpose and Scope	4
3. Logical Model	5
4. TWAMP Extensions	6
4.1. TWAMP-Control Extensions	7
4.1.1. Connection Setup with Services KPIs Monitoring	7
4.1.2. Services KPI-Monitor-REQ Command	7
4.1.3. Services KPI-Monitor-RSP Command	8
4.1.4. Services KPI-Monitor-IND Command	9
4.1.5. Services KPI-Monitor-ACK Command	10
4.1.6. Request-TW-Session Command Format	11
4.2. TWAMP-Test Extension	12
5. Acknowledgements	17
6. IANA Considerations	17
7. Security Considerations	19
8. Normative References	19
Authors' Addresses	19

1. Introduction

The TWAMP protocol defined in RFC 5357 [RFC5357] provides a common architecture for two way measurements in the IP network. In large scale IP environment, there are also large number of L4 to L7 services being deployed. Measuring these service performance become important when we measure the whole IP network performance. However the existing TWAMP protocol does not provide such capability which allow interaction between service plane and IP forwarding plane and monitor service KPI in the IP network.

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. In this document, we extend TWAMP protocol to calculate services KPIs and metrics in the network.

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 discussed in a separate draft 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 Server 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. This pair of two 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.

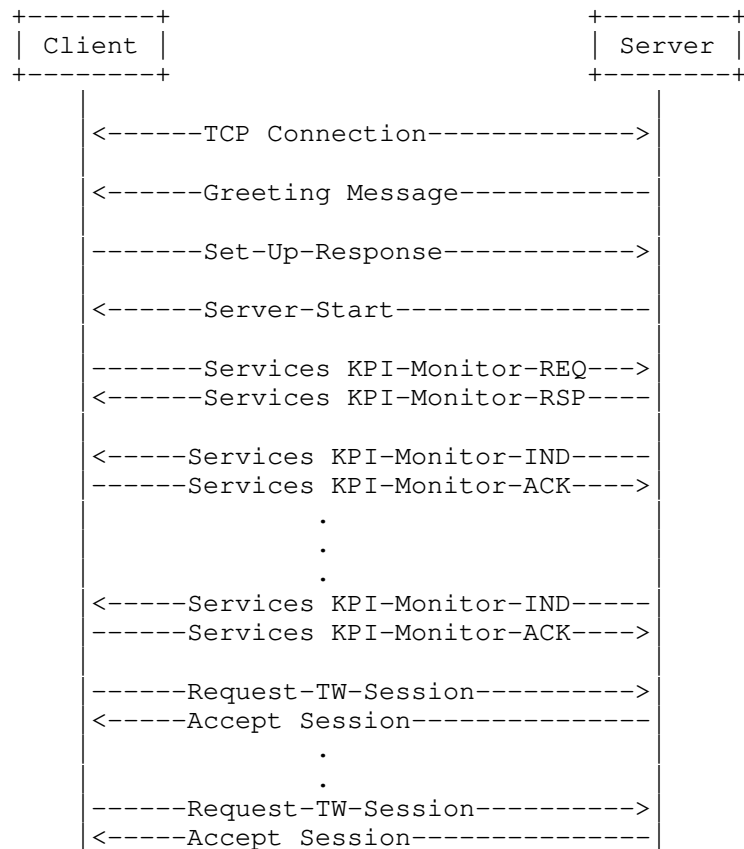


Figure 1

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:

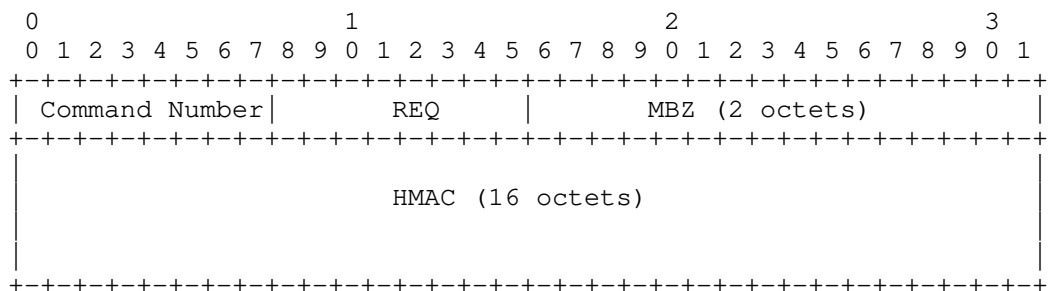


Figure 2: Services KPI-Monitor-REQ Command

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 corresponding 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:

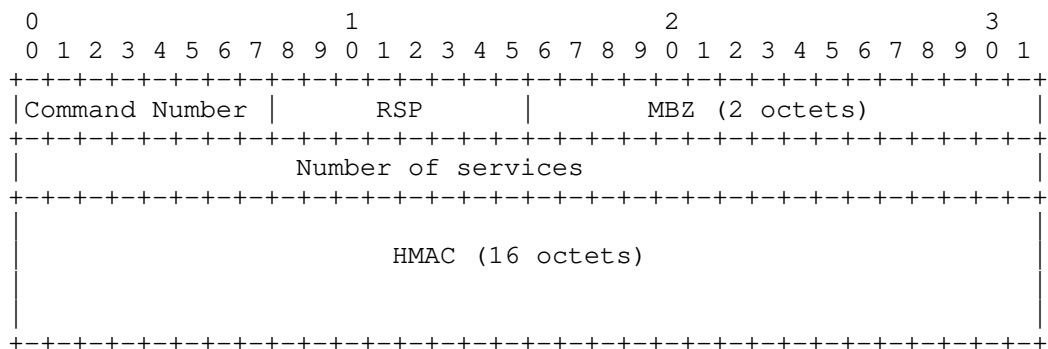


Figure 3: Services KPI-Monitor-RSP Command

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.

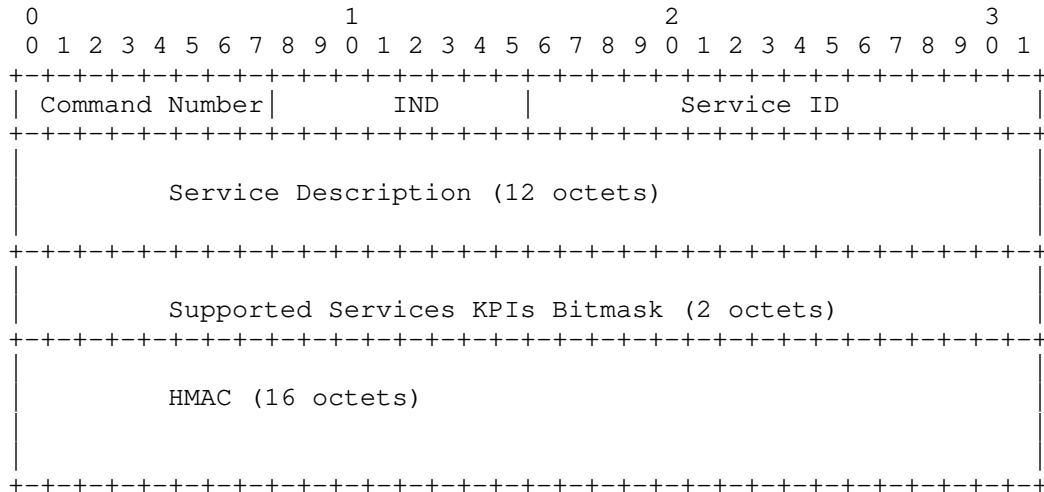


Figure 4: Services KPI-Monitor-IND Command

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.

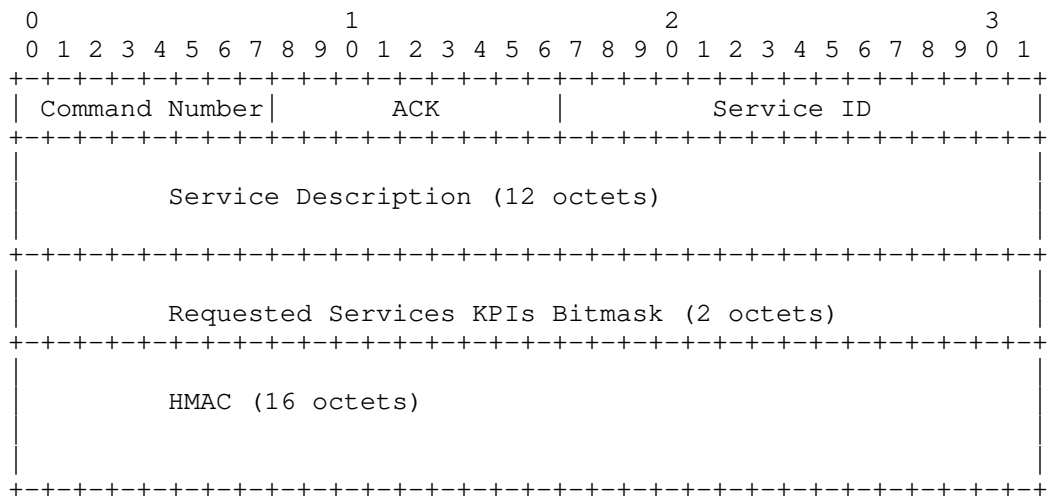


Figure 5: Services KPI-Monitor-ACK Command

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.

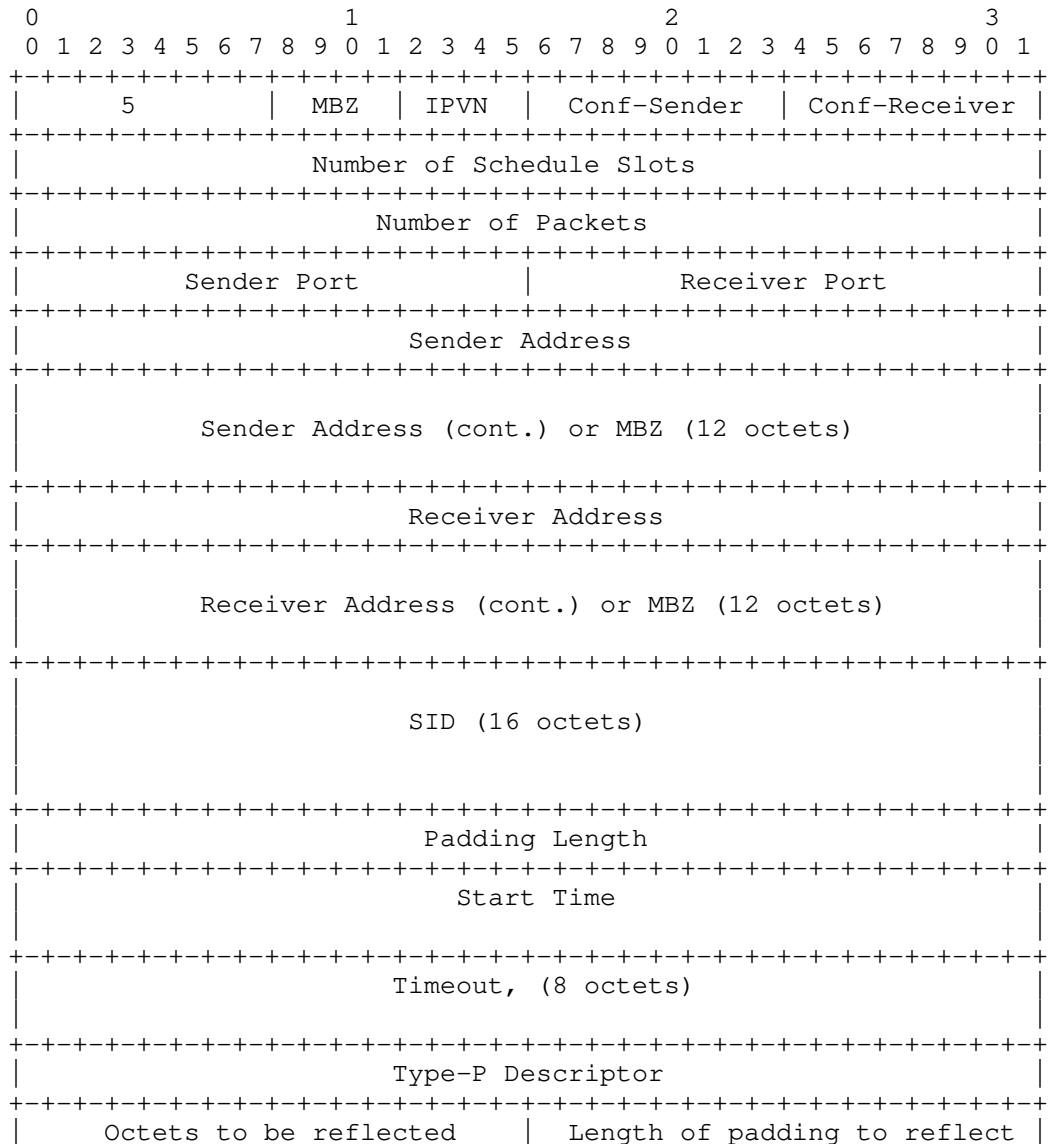
Service ID and Service Description MUST be same as that received in the Services KPI-Monitor-IND Command. These two fields together identify a particular Service.

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



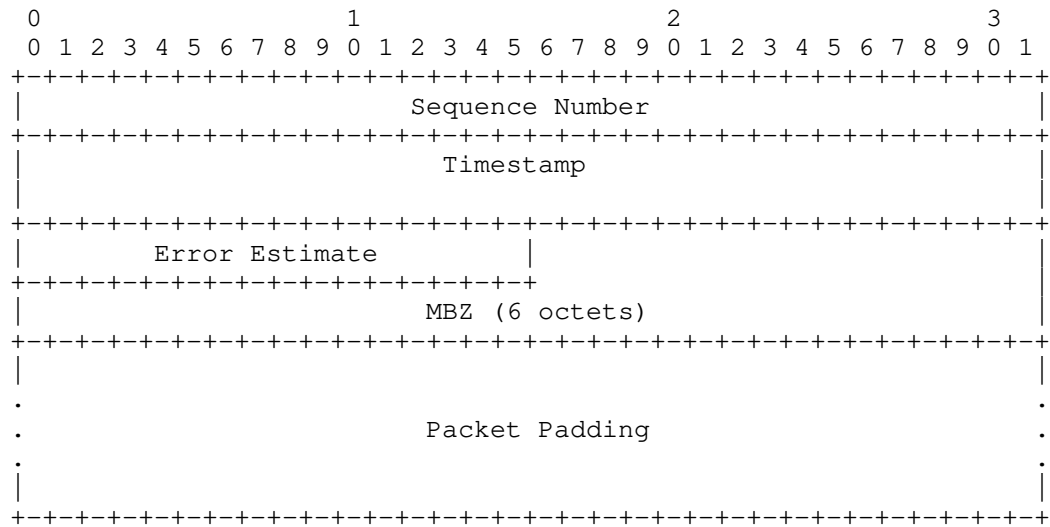


Figure 7: Unauthenticated Mode Session-Sender Data Packet Format

As a part of the extension, 6 octets of MBZ are added after the Error Estimate field.

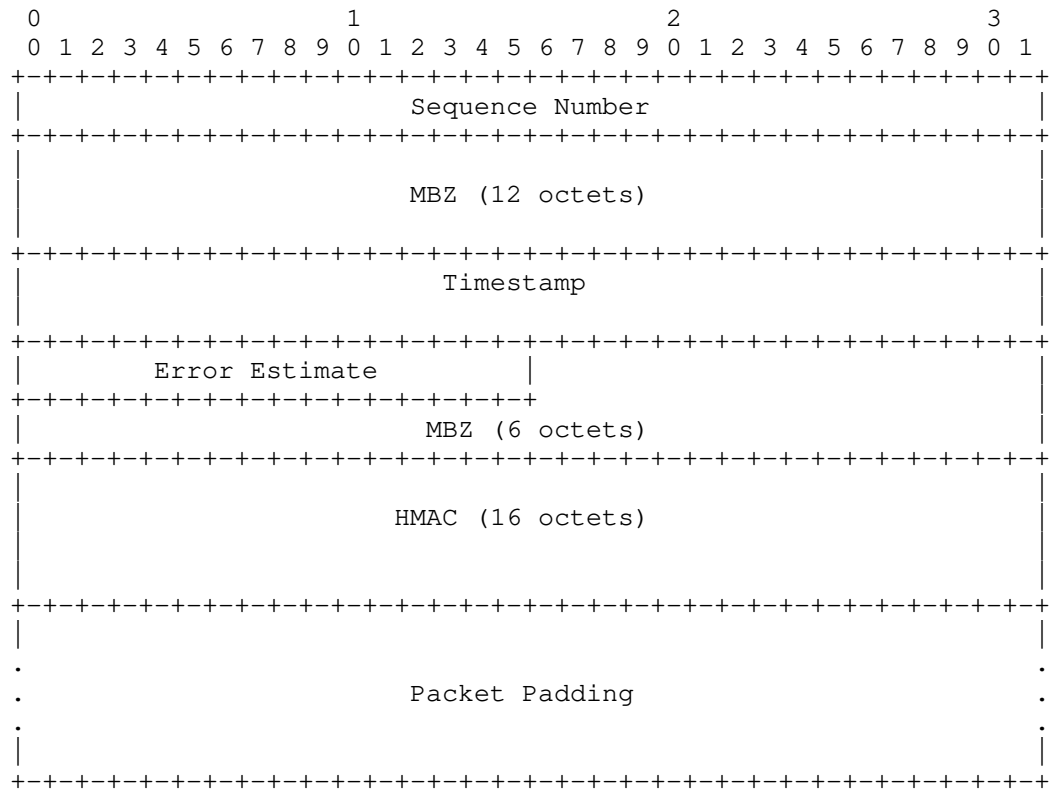


Figure 8: Authenticated and Encrypted Mode Session-Reflector Data Packet Format

The Session-Reflector Test Session data packet formats are provided below.

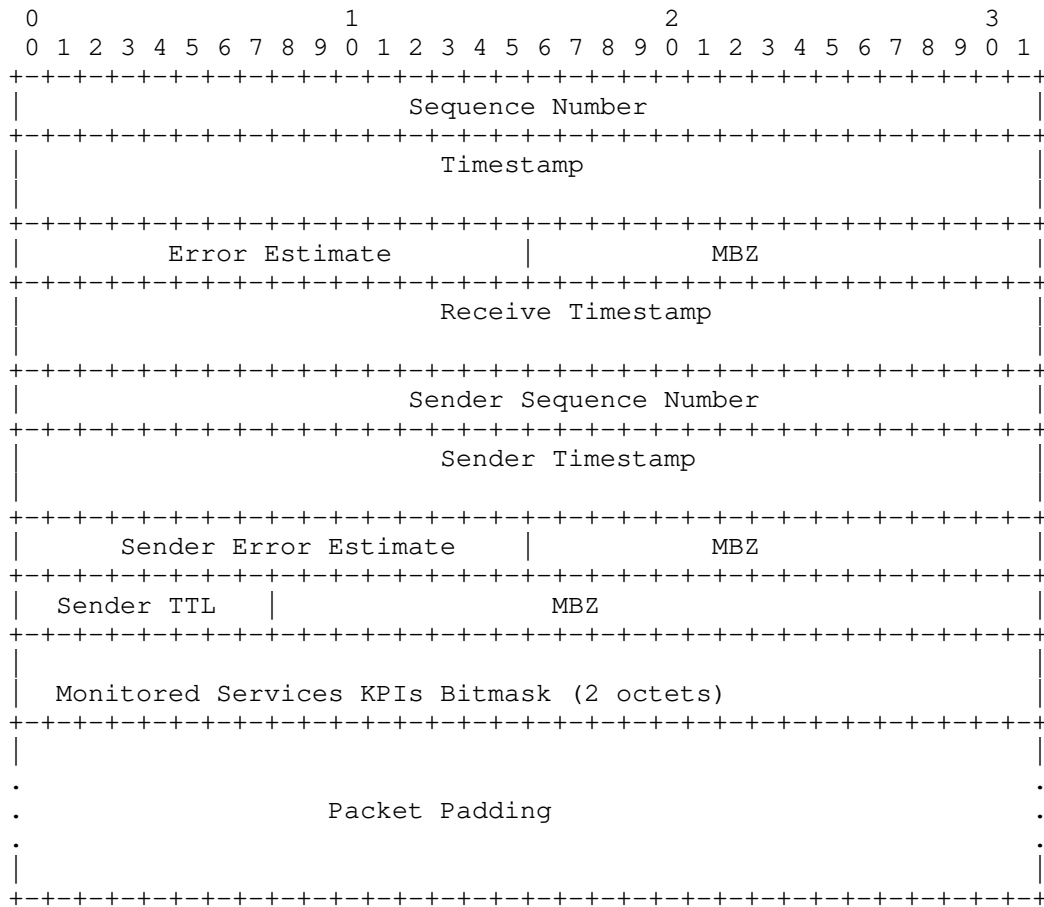
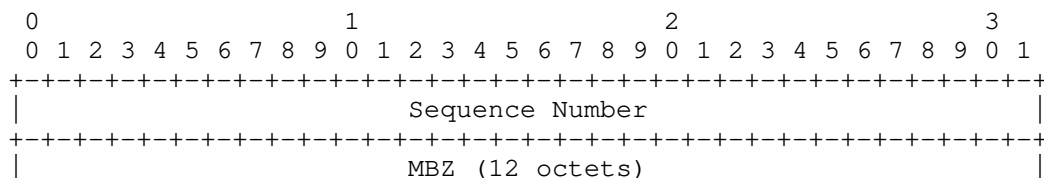


Figure 9: Unauthenticated Mode Session-Reflector Data Packet Format

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.



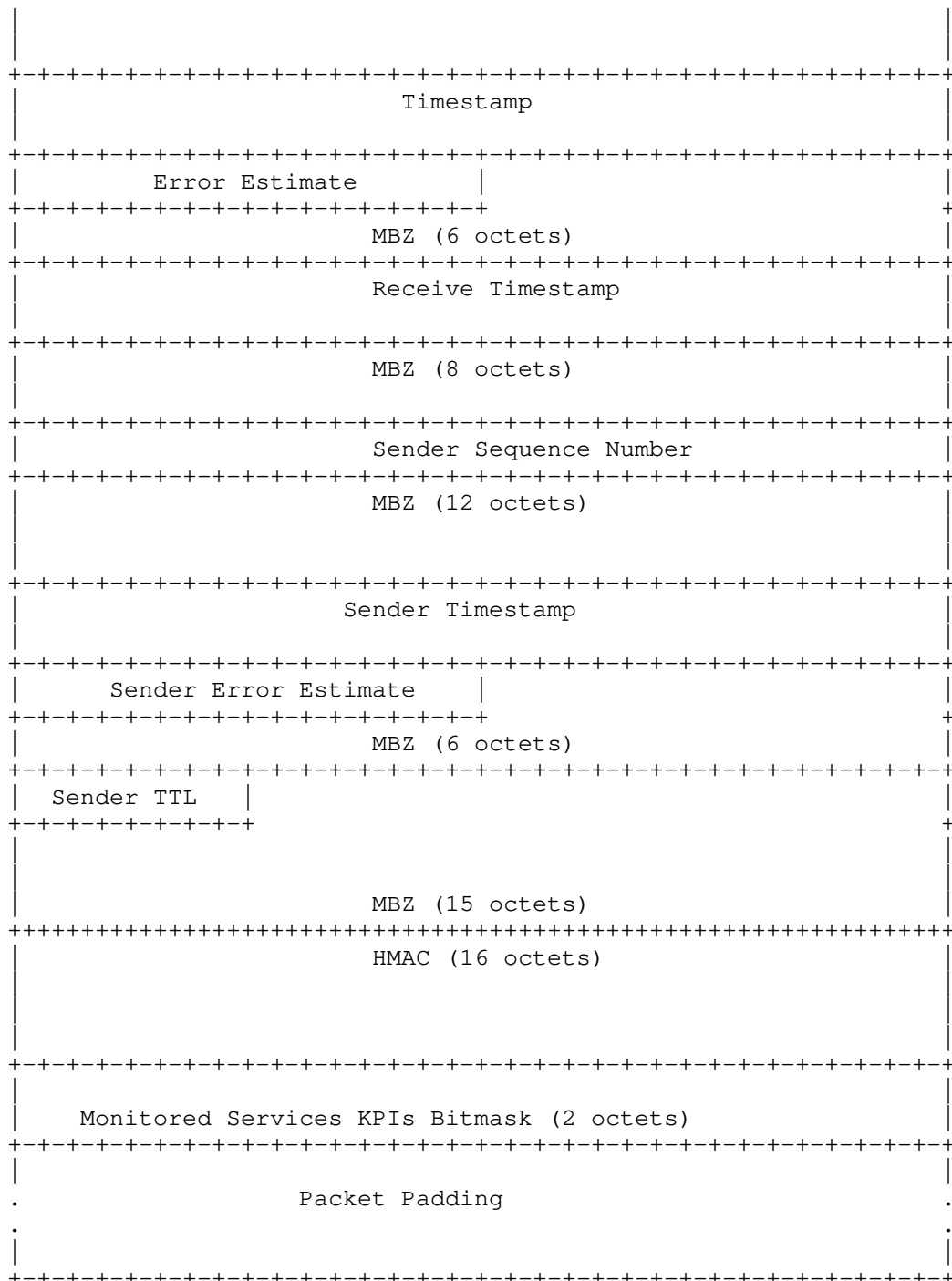


Figure 10: Authenticated and Encrypted Mode Session-Reflector Data Packet Format

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

The TWAMP protocol (RFC 5357) supports authenticated and encrypted mode for TWAMP session and data. The new extension proposed in this draft supports the authenticated and encrypted mode and is therefore provides a secure mechanism to monitor services KPIs

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
89, Asthagrama Layout 2nd Stage, Basavehwaranagar
Bangalore 560079
INDIA

Phone: +91 9845052354
Email: srivathsas@juniper.net

Peyush Gupta
Juniper Networks
Flat #206, 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

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Phone: +86-25-84565892
Email: bill.wu@huawei.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 22, 2016

A. Capello
M. Cociglio
G. Fioccola
L. Castaldelli
Telecom Italia
A. Tempia Bonda
March 21, 2016

A packet based method for passive performance monitoring
draft-tempia-ippm-p3m-03

Abstract

This document describes a passive method to perform packet loss, delay and jitter measurements on live traffic. This method is based on Alternate Marking (Coloring) technique. A report on the operational experiment done at Telecom Italia is explained in order to give an example and show the method applicability. This technique can be applied in various situations as detailed in this document. The previous IETF drafts about this technique were: [I-D.cociglio-mboned-multicast-pm] and [I-D.tempia-opsawg-p3m].

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 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	2
2. Overview of the method	4
3. Detailed description of the method	5
3.1. Packet loss measurement	5
3.2. One-way delay measurement	9
3.2.1. Single marking methodology	9
3.2.2. Average delay	11
3.2.3. Double marking methodology	11
3.3. Delay variation measurement	12
4. Implementation and deployment	12
4.1. Report on the operational experiment at Telecom Italia	13
4.1.1. Coloring the packets	14
4.1.2. Counting the packets	15
4.1.3. Collecting data and calculating packet loss	16
4.1.4. Metric transparency	17
4.2. IP flow performance measurement (IPFPM)	17
4.3. Performance Measurement Marking Method in BIER Domain	17
4.4. RFC6374 Use Case	17
4.5. Application to active performance measurement	18
5. Hybrid measurement	18
6. Compliance with RFC6390 guidelines	18
7. Security Considerations	20
8. Conclusions	20
9. IANA Considerations	21
10. Acknowledgements	21
11. References	21
11.1. Normative References	21
11.2. Informative References	22
Authors' Addresses	23

1. Introduction

Nowadays, most of the traffic in Service Providers' networks carries real time content. These contents are highly sensitive to packet loss [RFC2680], while interactive contents are sensitive to delay [RFC2679], and jitter [RFC3393].

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

management (e.g. isolation of network problems, troubleshooting, etc.).

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

The lack of adequate tools to measure packet loss with the desired accuracy drove an effort to design a new method for the performance monitoring of live traffic, possibly easy to implement and deploy. The effort led to the method described in this document: basically, it is a passive performance monitoring technique, potentially applicable to any kind of packet based traffic, including Ethernet, IP, and MPLS, both unicast and multicast. The method addresses primarily packet loss measurement, but it can be easily extended to one-way delay and delay variation measurements as well. It doesn't require any protocol extension or interaction with existing protocols, thus avoiding any interoperability issue. Even if the method doesn't raise any specific need for standardization, it could be further improved by means of some extension to existing protocols, but this aspect is left for further study and it is out of the scope of this document.

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

The method described in this document, also called PNPM (Packet Network Performance Monitoring), has been invented and engineered in Telecom Italia and it's currently being used in Telecom Italia's network. The previous IETF drafts about this technique were: [I-D.cociglio-mboned-multicast-pm] and [I-D.tempia-opsawg-p3m]. There are some references to this methodology in other IETF works (e.g. [I-D.ietf-mpls-flow-ident], [I-D.bryant-mpls-sfl-framework] [I-D.bryant-mpls-rfc6374-sfl], [I-D.ietf-bier-mpls-encapsulation], [I-D.mirsky-bier-pmmm-oam] [I-D.chen-ippm-coloring-based-ipfpm-framework]).

This document is organized as follows:

- o Section 2 gives an overview of the method, including a comparison with different measurement strategies;
- o Section 3 describes the method in detail;
- o Section 4 reports examples of implementation and deployment of the method. Furthermore the operational experiment done at Telecom Italia is described;
- o Section 5 includes some considerations about security aspects;
- o Section 6 finally summarizes some concluding remarks.

2. Overview of the method

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

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

The method proposed in this document follows the second approach, but it doesn't use additional packets to virtually split the flow in blocks. Instead, it "colors" the packets so that the packets belonging to the same block will have the same color, whilst consecutive blocks will have different colors. Each change of color represents a sort of auto-synchronization signal that guarantees the consistency of measurements taken by different devices along the path.

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

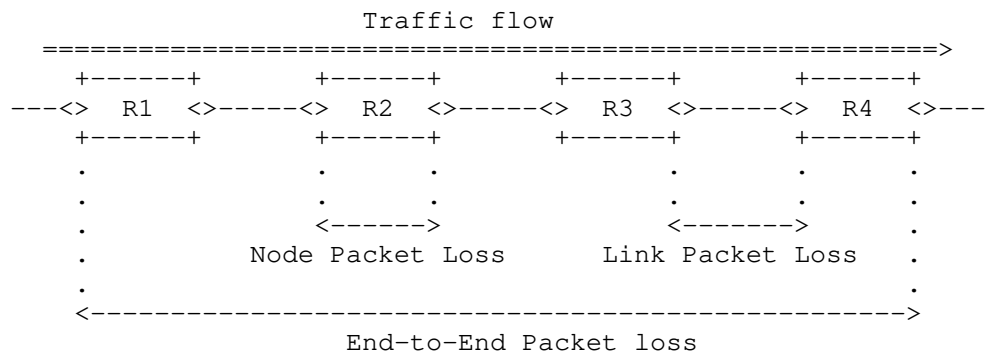


Figure 1: Available measurements

3. Detailed description of the method

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

3.1. Packet loss measurement

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

As discussed in the previous section, a simple way to create the blocks is to "color" the traffic (two colors are sufficient) so that packets belonging to different consecutive blocks will have different colors. Whenever the color changes, the previous block terminates and the new one begins. Hence, all the packets belonging to the same block will have the same color and packets of different consecutive blocks will have different colors. The number of packets in each block depends on the criterion used to create the blocks: if the color is switched after a fixed number of packets, then each block will contain the same number of packets (except for any losses); but if the color is switched according to a fixed timer, then the number of packets may be different in each block depending on the packet rate.

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

A: packet with A coloring

B: packet with B coloring

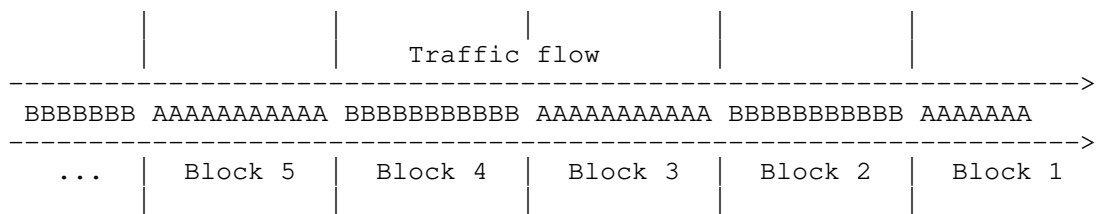


Figure 2: Traffic coloring

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

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

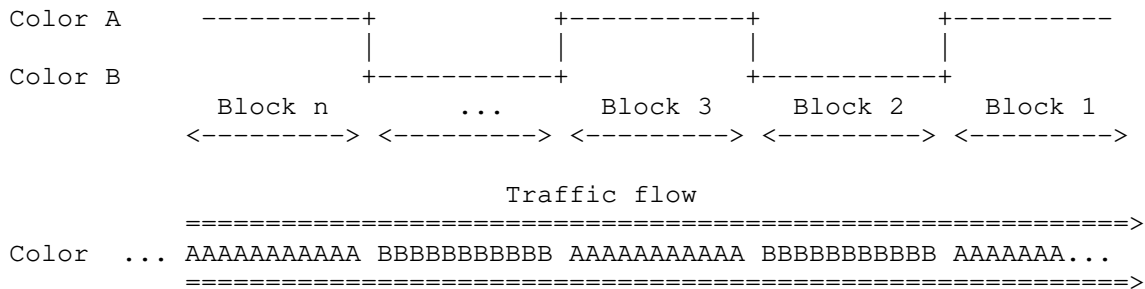


Figure 3: Application of the method to compute link packet loss

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

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

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

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

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

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

Table 1: Evaluation of counters for packet loss measurements

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

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

For example, looking at the table above, during the first block (A-colored), C(A)R1 and C(A)R2 have the same value (375), which

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

R1 and R2 require a clock error less than $\pm L/2$ time units, where L is the time duration of the block. In this way each colored packet can be assigned to the right block by each router. This is because the minimum time distance between two packets of the same color but belonging to different blocks is L time units.

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

3.2. One-way delay measurement

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

3.2.1. Single marking methodology

The alternation of colors can be used as a time reference to calculate the delay. Whenever the color changes (that means that a new block has started) a network device can store the timestamp of the first packet of the new block; that timestamp can be compared with the timestamp of the same packet on a second router to compute packet delay. Considering Figure 4, R1 stores a timestamp TS(A1)R1 when it sends the first packet of block 1 (A-colored), a timestamp TS(B2)R1 when it sends the first packet of block 2 (B-colored) and so on for every other block. R2 performs the same operation on the receiving side, recording TS(A1)R2, TS(B2)R2 and so on. Since the timestamps refer to specific packets (the first packet of each block) we are sure that timestamps compared to compute delay refer to the same packets. By comparing TS(A1)R1 with TS(A1)R2 (and similarly TS(B2)R1 with TS(B2)R2 and so on) it is possible to measure the delay between R1 and R2. In order to have more measurements, it is possible to take and store more timestamps, referring to other packets within each block.

In order to coherently compare timestamps collected on different routers, the network nodes must be in sync. Furthermore, a measurement is valid only if no packet loss occurs and if packet misordering can be avoided, otherwise the first packet of a block on

R1 could be different from the first packet of the same block on R2 (f.i. if that packet is lost between R1 and R2 or it arrives after the next one).

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

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

Table 2: Evaluation of timestamps for delay measurements

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

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

corresponding timestamps (possible but impractical from an implementation point of view).

3.2.2. Average delay

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

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

3.2.3. Double marking methodology

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

Basically, the idea is to use the first marking to create the alternate flow and, within this colored flow, a second marking to select the packets for measuring delay/jitter. The first marking is

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

3.3. Delay variation measurement

Similarly to one-way delay measurement (both for single marking and double marking), the method can also be used to measure the inter-arrival jitter. The alternation of colors can be used as a time reference to measure delay variations. Considering the example depicted in Figure 4, R1 stores a timestamp TS(A)R1 whenever it sends the first packet of a block and R2 stores a timestamp TS(B)R2 whenever it receives the first packet of a block. The inter-arrival jitter can be easily derived from one-way delay measurement, by evaluating the delay variation of consecutive samples.

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

4. Implementation and deployment

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

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

4.1. Report on the operational experiment at Telecom Italia

The methodology has been applied in Telecom Italia by leveraging functions and tools available on IP routers and it's currently being used to monitor packet loss in some portions of Telecom Italia's network. The application of the method to delay measurement is currently being evaluated in Telecom Italia's labs. This section describes how the features currently available on existing routing platforms can be used to apply the method, in order to give an example of implementation and deployment.

The fundamental steps for this implementation of the method can be summarized in the following items:

- o coloring the packets;
- o counting the packets;
- o collecting data and calculating the packet loss.
- o metric transparency.

Before going deeper into the implementation details, it's worth mentioning two different strategies that can be used when implementing the method:

- o flow-based: the flow-based strategy is used when only a limited number of traffic flows need to be monitored. This could be the case, for example, of IPTV channels or other specific applications traffic with high QoS requirements (i.e. Mobile Backhauling traffic). According to this strategy, only a subset of the flows is colored. Counters for packet loss measurements can be instantiated for each single flow, or for the set as a whole, depending on the desired granularity. A relevant problem with this approach is the necessity to know in advance the path followed by flows that are subject to measurement. Path rerouting and traffic load-balancing increase the issue complexity, especially for unicast traffic. The problem is easier to solve for multicast traffic where load balancing is seldom used, especially for IPTV traffic where static joins are frequently used to force traffic forwarding and replication. Another application is on Mobile Backhauling, implemented with a VPN MPLS in Telecom Italia's network; in this case the problem with unicast traffic is overcome by monitoring just the two Provider Edge nodes of the VPN MPLS.
- o link-based: measurements are performed on all the traffic on a link by link basis. The link could be a physical link or a

logical link (for instance an Ethernet VLAN or a MPLS PW). Counters could be instantiated for the traffic as a whole or for each traffic class (in case it is desired to monitor each class separately), but in the second case a couple of counters is needed for each class.

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

4.1.1. Coloring the packets

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

In case of flow-based measurements, it is desirable, in general, to have a single coloring node because it is easier to manage and doesn't rise any risk of conflict (consider the case where two nodes color the same flow). Thus it is necessary to color the flow as close as possible to the source. In addition, coloring a flow close to the source allows an end-to-end measure if a measurement point is enabled on the last-hop router as well. The only requirement is that the coloring must change periodically and every node along the path must be able to identify unambiguously the colored packets. For link-based measurements, all traffic needs to be colored when

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

Traffic coloring can be implemented by setting a specific bit in the packet header and changing the value of that bit periodically. With current router implementations, only QoS related fields and features offer the required flexibility to set bits in the packet header. In case a Service Provider only uses the three most significant bits of the DSCP field (corresponding to IP Precedence) for QoS classification and queuing, it is possible to use the two less significant bits of the DSCP field (bit 0 and bit 1) to implement the method without affecting QoS policies. One of the two bits (bit 0) could be used to identify flows subject to traffic monitoring (set to 1 if the flow is under monitoring, otherwise it is set to 0), while the second (bit 1) can be used for coloring the traffic (switching between values 0 and 1, corresponding to color A and B) and creating the blocks.

In practice, coloring the traffic using the DSCP field can be implemented by configuring on the router output interface an access list that intercepts the flow(s) to be monitored and applies to them a policy that sets the DSCP field accordingly. Since traffic coloring has to be switched between the two values over time, the policy needs to be modified periodically: an automatic script can be used to perform this task on the basis of a fixed timer. In Telecom Italia's implementation this timer is set to 5 minutes: this value showed to be a good compromise between measurement frequency and stability of the measurement (i.e. possibility to collect all the measures referring to the same block).

4.1.2. Counting the packets

Assuming that the coloring of the packets is performed only by the source node, the nodes between source and destination (included) have to count the colored packets that they receive and forward: this operation can be enabled on every router along the path or only on a subset, depending on which network segment is being monitored (a single link, a particular metro area, the backbone, the whole path).

Since the color switches periodically between two values, two counters (one for each value) are needed: one counter for packets with color A and one counter for packets with color B. For each flow (or group of flows) being monitored and for every interface where the monitoring is active, a couple of counters is needed. For example, in order to monitor separately 3 flows on a router with 4 interfaces

involved, 24 counters are needed (2 counters for each of the 3 flows on each of the 4 interfaces). If traffic is colored using the DSCP field, as in Telecom Italia's implementation, an access-list that matches specific DSCP values can be used to count the packets of the flow(s) being monitored.

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

Another important aspect to take into consideration is when to read the counters: in order to count the exact number of packets of a block the routers must perform this operation when that block has ended: in other words, the counter for color A must be read when the current block has color B, in order to be sure that the value of the counter is stable. This task can be accomplished in two ways. The general approach suggests to read the counters periodically, many times during a block duration, and to compare these successive readings: when the counter stops incrementing means that the current block has ended and its value can be elaborated safely. Alternatively, if the coloring operation is performed on the basis of a fixed timer, it is possible to configure the reading of the counters according to that timer: for example, if each block is 5 minutes long, reading the counter for color A every 5 minute in the middle of the subsequent block (with color B) is a safe choice. A sufficient margin should be considered between the end of a block and the reading of the counter, in order to take into account any out-of-order packets. The choice of a 5 minutes timer for colore switching was also inspired by these considerations.

4.1.3. Collecting data and calculating packet loss

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

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

values from all the routers within one cycle of the timer (5 minutes).

If link-based measurement is used, it would be possible to use a protocol to exchange values of counters between the two endpoints in order to let them perform the packet loss calculation for each traffic direction. A similar approach could be complicated if applied to a flow-based measurement.

4.1.4. Metric transparency

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

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

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

4.2. IP flow performance measurement (IPFPM)

This application of marking method is described in [I-D.chen-ippm-coloring-based-ipfpm-framework].

4.3. Performance Measurement Marking Method in BIER Domain

In [I-D.ietf-bier-mpls-encapsulation] two OAM bits from Bit Index Explicit Replication (BIER) Header are reserved for the passive performance measurement marking method. [I-D.mirsky-bier-pmmm-oam] details the measurement for multicast service over BIER domain.

4.4. RFC6374 Use Case

RFC6374 [RFC6374] uses the LM packet as the packet accounting demarcation point. Unfortunately this gives rise to a number of problems that may lead to significant packet accounting errors in certain situations. [I-D.ietf-mpls-flow-ident] discusses the desired

capabilities for MPLS flow identification in order to perform a better in-band performance monitoring of user data packets. A method of accomplishing identification is Synonymous Flow Labels (SFL) introduced in [I-D.bryant-mpls-sfl-framework], while [I-D.bryant-mpls-rfc6374-sfl] describes RFC6374 performance measurements with SFL.

4.5. Application to active performance measurement

[I-D.fioccola-ippm-rfc6812-alt-mark-ext] describes an extension to the Cisco SLA Protocol Measurement-Type UDP-Measurement, in order to implement alternate marking methodology.

5. Hybrid measurement

The method has been explicitly designed for passive measurements but it can also be used with active measurements. In order to have both end to end measurements and intermediate measurements (hybrid measurements) two end points can exchanges artificial traffic flows and apply alternate marking over these flows. In the intermediate points artificial traffic is managed in the same way as real traffic and measured as specified before.

6. Compliance with RFC6390 guidelines

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

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

- o Metric name and description: as already stated, this document doesn't propose any new Performance Metric. On the contrary, it describes a novel method for measuring packet loss [RFC2680]. The same concept, with small differences, can also be used to measure delay [RFC2679], and jitter [RFC3393]. The document mainly describes the applicability to packet loss measurement.
- o Method of Measurement or Calculation: according to the method described in the previous sections, the number of packets lost is calculated by subtracting the value of the counter on the source

node from the value of the counter on the destination node. Both counters must refer to the same color. The calculation is performed when the value of the counters is in a steady state.

- o Units of Measurement: the method calculates and reports the exact number of packets sent by the source node and not received by the destination node.
- o Measurement Points: the measurement can be performed between adjacent nodes, on a per-link basis, or along a multi-hop path, provided that the traffic under measurement follows that path. In case of a multi-hop path, the measurements can be performed both end-to-end and hop-by-hop.
- o Measurement Timing: the method have a constraint on the frequency of measurements. In order to perform a measure, the counter must be in a steady state: this happens when the traffic is being colored with the alternate color; for example in the Telecom Italia application of the method the time interval is set to 5 minutes.
- o Implementation: the Telecom Italia application of the method uses two encodings of the DSCP field to color the packets; this enables the use of policy configurations on the router to color the packets and accordingly configure the counter for each color. The path followed by traffic being measured should be known in advance in order to configure the counters along the path and be able to compare the correct values.
- o Use and Applications: the method can be used to measure packet loss with high precision on live traffic; moreover, by combining end-to-end and per-link measurements, the method is useful to pinpoint the single link that is experiencing loss events.
- o Reporting Model: the value of the counters has to be sent to a centralized management system that perform the calculations; such samples must contain a reference to the time interval they refer to, so that the management system can perform the correct correlation; the samples have to be sent while the corresponding counter is in a steady state (within a time interval), otherwise the value of the sample should be stored locally.
- o Dependencies: the values of the counters have to be correlated to the time interval they refer to; moreover, as far the Telecom Italia application of the method is based on DSCP values, there are significant dependencies on the usage of the DSCP field: it must be possible to rely on unused DSCP values without affecting QoS-related configuration and behavior; moreover, the intermediate

nodes must not change the value of the DSCP field not to alter the measurement.

- o Organization of Results: the method of measurement produces singletons.
- o Parameters: currently, the main parameter of the method is the time interval used to alternate the colors and read the counters.

7. Security Considerations

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

There are two types of security concerns: potential harm caused by the measurements and potential harm to the measurements. For what concerns the first point, the measurements described in this document are passive, so there are no packets injected into the network causing potential harm to the network itself and to data traffic. Nevertheless, the method implies modifications on the fly to the IP header of data packets: this must be performed in a way that doesn't alter the quality of service experienced by packets subject to measurements and that preserve stability and performance of routers doing the measurements. The measurements themselves could be harmed by routers altering the coloring of the packets, or by an attacker injecting artificial traffic. Authentication techniques, such as digital signatures, may be used where appropriate to guard against injected traffic attacks.

The privacy concerns of network measurement are limited because the method only relies on information contained in the IP header without any release of user data.

8. Conclusions

The advantages of the method described in this document are:

- o easy implementation: it can be implemented using features already available on major routing platforms;
- o low computational effort: the additional load on processing is negligible;

- o accurate packet loss measurement: single packet loss granularity is achieved with a passive measurement;
- o potential applicability to any kind of packet/frame -based traffic: Ethernet, IP, MPLS, etc., both unicast and multicast;
- o robustness: the method can tolerate out of order packets and it's not based on "special" packets whose loss could have a negative impact;
- o no interoperability issues: the features required to implement the method are available on all current routing platforms.

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

9. IANA Considerations

There are no IANA actions required.

10. Acknowledgements

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

11. References

11.1. Normative References

- [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>>.
- [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>>.

11.2. Informative References

[I-D.bryant-mpls-rfc6374-sfl]

Bryant, S., Swallow, G., Sivabalan, S., Mirsky, G., Chen, M., and Z. Li, "RFC6374 Synonymous Flow Labels", draft-bryant-mpls-rfc6374-sfl-00 (work in progress), October 2015.

[I-D.bryant-mpls-sfl-framework]

Bryant, S., Swallow, G., Sivabalan, S., Mirsky, G., Chen, M., and Z. Li, "Synonymous Flow Label Framework", draft-bryant-mpls-sfl-framework-00 (work in progress), October 2015.

[I-D.bryant-mpls-synonymous-flow-labels]

Bryant, S., Swallow, G., Sivabalan, S., Mirsky, G., Chen, M., and Z. Li, "RFC6374 Synonymous Flow Labels", draft-bryant-mpls-synonymous-flow-labels-01 (work in progress), July 2015.

[I-D.chen-ippm-coloring-based-ipfpm-framework]

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

[I-D.cociglio-mboned-multicast-pm]

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

[I-D.fioccola-ippm-rfc6812-alt-mark-ext]

Fioccola, G., Clemm, A., Cociglio, M., Chandramouli, M., and A. Capello, "Alternate Marking Extension to Cisco SLA Protocol RFC6812", draft-fioccola-ippm-rfc6812-alt-mark-ext-00 (work in progress), October 2015.

[I-D.ietf-bier-mpls-encapsulation]

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

[I-D.ietf-mpls-flow-ident]

Bryant, S., Pignataro, C., Chen, M., Li, Z., and G. Mirsky, "MPLS Flow Identification Considerations", draft-ietf-mpls-flow-ident-00 (work in progress), December 2015.

[I-D.mirsky-bier-pmmm-oam]

Mirsky, G., Zheng, L., Chen, M., and G. Fioccola,
"Performance Measurement (PM) with Marking Method in Bit
Index Explicit Replication (BIER) Layer", draft-mirsky-
bier-pmmm-oam-01 (work in progress), March 2016.

[I-D.tempia-opsawg-p3m]

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

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

[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>>.

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

Authors' Addresses

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

Email: alessandro.capello@telecomitalia.it

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

Email: mauro.cociglio@telecomitalia.it

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

Email: giuseppe.fioccola@telecomitalia.it

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

Email: luca.castaldelli@telecomitalia.it

Alberto Tempia Bonda

Email: alberto.tempia@gmail.com