IPPM Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2017

B. M Gaonkar
S. Jacob
Juniper
G. Fioccola
Telecom Italia
Q. Wu
Huawei
P. Ananthasankaran
Nokia
March 5, 2017

Packet Loss measurement Model
draft-bhaprasud-ippm-pm-02

Abstract

This document defines the loss measurement matrix models for service level packets on the network which can be implemented in different kind of network scenarios.

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 6, 2017.

Copyright Notice

Table of Contents

1. Introduction

Today, Performance monitoring or tracking of the performance experienced by customer traffic is a key technology to strengthen service offering based on enhanced QoE and SLAs. The lack of adequate tools to detect an interesting subset of a Packet Stream, as identified by a particular packet attribute(e.g., commit rate or DSCP) and measure that packet loss drives an effort to design a new method for the performance monitoring of live traffic, possibly easy to implement and deploy. The draft aims to define loss measurement matrix models for multiple customer service flows on the network. Each customer service flow is corresponding to an interesting subset of the same packet stream. The customer or packet stream can be identified by a list of source or destination prefixes, or by ingress or egress interfaces.

The network would be provisioned with multiple services(e.g., real time service, interactive service) having different SLAs(e.g., bandwidth constraint or end packet loss constraint for the end to end path) based on the customers' requirement. This models aims at

computing Loss measurement for these services (belonging to the same customer)independently for each defined SLA matrixes.

The class-of-service and packet color classification defined in the network is a key factor to classify network traffic and drive traffic management mechanism to achieve corresponding SLA for each service. This draft uses the class-of-service model and color based model for any given network to define the packet loss measurement for various services with the different SLA requirements.

The proposed matrix models is suitable mainly for passive performance measurements but can be considered for active and hybrid performance measurements as well.

This solution models loss measurement in different kinds of network scenarios. The different models explained here will help to analyse packet loss pattern, analyze the network congestion in a better way and model the network in a better way. Loss measurement is carried out between 2 end points.The underlying technology could be an active loss measurement or a Passive loss measurement.

Any loss measurement will require 2 counters:

- Number of packets transmitted from one end point.
- Number of packets received at the other end point.

This draft explains the different ways to model the above data and get meaningful result for the loss measurement compulation. The underlying technology could be an MPLS Loss measurement, or based loss measurement or an IP based loss measurement.

2. Conventions used in this document

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 [RFC2119].

Observation Point An Observation Point is a location in the network where data packets can be observed. Examples include a line to which a probe is attached, a shared medium, such as an Ethernet-based LAN, a single port of a router, or a set of interfaces (physical or logical) of a router.

Persistence Data Store The persistence Data store is a scalable data store which collects time based data such as streaming data or time series data for network analytics.

Time Series Data Time Series Data is a sequence of data points with time stamps. The data points are limited to loss measurement results in this document.

Packet Stream A Packet Stream denotes a set of packets from the Observed Packet Stream that flows past some specified point within the Metering Process. An example of a Packet Stream is the output of the Selection Process.

Packet Content The Packet Content denotes the union of the packet header (which includes link layer, network layer, and other encapsulation headers) and the packet payload.

Color Identifier: It is used to identify the color that applies to the data packet. Color identifier can be assigned to service level packet based on commit rate and excess rate set for the traffic. For example, the service level packet will be set with "green" color if it is less than committed" rate; the Service Level packet will be set with "yellow" color if it is exceeding the"committed" rate but less than the "excess" rate. The service frame will be set with "red" color if it is exceeding both the "committed" and "excess" rates.

COS Identifier: It is used to identify the COS that applies to the data packet.CoS identifier can be assigned based on dot1p value in C-tag, or DSCP in IP header.

Complete data measurement: Complete data measurement is a data measurement method which monitors every packet and condense a large amount of information about packet arrivals into a small number of statistics. The aim of "monitoring every packet" is to ensure that the information reported is not dependent on the application.

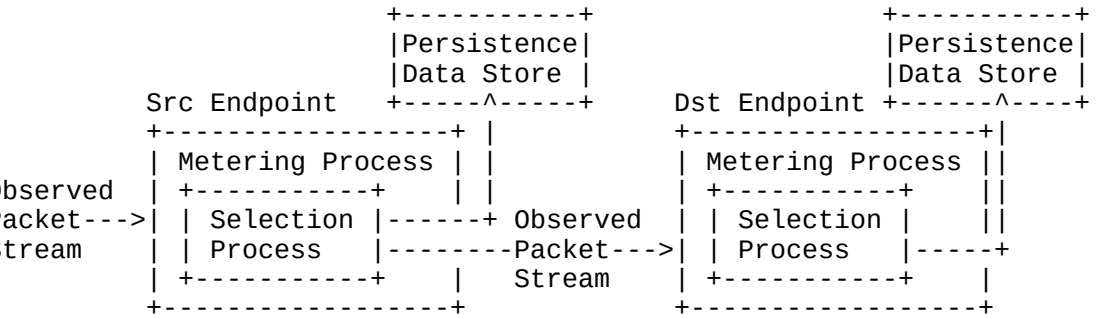
Color based data measurement: Color based data measurement is a data measurement method which monitors the data packet with the same color identifier. COS identifier could be C-Tag Priority Code Point(PCP) or DSCP.

COS and color based Data measurement: COS and color based Data measurement is a data measurement method which monitors the data packet with the same defined SLA matrix.The SLA matrix is an array of Color identifier attribute and COS identifier attribute.

3. Traffic Management Architecture

A stream of packets is observed at an Observation Point of the source endpoint and destination endpoints. Two observation points can also be placed at the same endpoint for node monitoring [I-D.ietf-ippm-alt-mark], i.e.,one is at ingress interface of the endpoint and the other is at the egress interface of the endpoint. A Selection Process inspects each packet to determine whether or not it is to be selected for data analytics. The Selection Process is part of the Metering Process, which constructs a report stream on selected packets as output, using the Packet Content, and possibly other information such as the arrival timestamp. The report stream on selected packets will be stored in the persistence data store for real time data analysis or time sequence data analysis.

The following figure indicates the sequence of the three processes (Selection, Metering, and Storing).



3.1. Selection Process

This section defines the Selection Process and related objects.

Selection Process: A Selection Process takes the Observed Packet Stream as its input and selects a subset of that stream as its output.

Selection State: A Selection Process may maintain state information for use by the Selection Process. At a given time, the Selection State may depend on packets observed at and before that time, and other variables. Examples include sequence numbers of packets at the input of Selectors,a timestamp of observation of the packet at the Observation Point,indicators of whether the packet was selected by a given Selector.

Selector: A Selector defines the action of a Selection Process on a single packet of its input. If selected, the packet becomes an element of the output Packet Stream.

The Selector can make use of the following information in determining whether a packet is selected:

* COS Identifier in the Packet Content;

* Traffic attribute such as Color identifier;

* Combination of CoS Identifier and Color Identifier

3.2. Metering Process

A Metering Process selects packets from the Observed Packet Stream using a Selection Process, and produces as output a Report Stream concerning the selected packets.

4. Loss Measurement Models

4.1. Complete data measurement (Monitoring all the traffic)

This model uses the complete data traffic between the 2 end-points to compute loss measurement. This will result in computation of loss measurement for the entire traffic in the network in one direction. This is primarily used in cases of backbone traffic where traffic from different services are aggregated and send into the core network.This will count all the packet, this gives the overall loss measurment between one endpoint to other.

4.2. Color based data measurement

This is same as the above section of "complete data measurement" with a minor difference, only monitoring the data packet with specific color identifier.

In this model the packets are counted in the following Way: Count specific data traffic with different color identifier between 2 end points for loss measurement.One example of Color based data measurement is to count two type of color based traffic:

Count all committed traffic between the 2 end-point for loss measurement.

Count all Excess traffic which is beyond the committed traffic for the specific network.

When both of these are combined then it becomes the model for complete traffic as mentioned in the above section.

In practice the Color of traffic can be using any mechanism based on the network encapsulation.As long as the packets could be treated differently based on the underlying encapsulation this mechanism could be used.

This is used in core networks where the aggregated traffic has differential priority and loss measurement can be computed on the committed traffic which is guaranteed in the network when compared with excess traffic which could be dropped based on network load and provisioning.

4.3. COS based Data measurement

This model uses the data traffic in the network which is flowing in a specific COS to measure the loss in the network.Based on the class of traffic in the network the transmitted and received packets are counted to calculate the loss measurement.

Primary use of this kind of loss measurement is to measure loss measurement for a specific service which has strict SLAs. The service could be a point-to-point layer2 service, an MPLS based service.

4.4. COS and color based Data measurement

This model uses a combination of both Color based data measurement and Cos based data measurement. Packets are counter for a specific COS with a specific color.This can count both in profile packet which are green and yellow which are out profile packets. This will not count the red packet which violates the SLA.This will count the packet for each SLA and color separately.

5. Active and Passive performance measurements

This model reinforces the use of well known methodologies for passive performance measurements.A very simple, flexible and straightforward mechanism is presented in [I-D.ietf-ippm-alt-mark]. The basic idea is to virtually split traffic flows into consecutive batches of packets:each block represents a measurable entity unambiguously recognizable thanks to the alternate marking. This approach, called Alternate Marking method, is efficient both for passive performance monitoring and for active performance monitoring.

6. Use Cases

Consider a provider running point to point service between router A and B for his customer "X".Customer "X" has voice traffic which requires special treatment,then he requires attention for database traffic. The customer "X" has SLA with the provider. Now the challenge faced by the provider is how to measure the traffic of customer "X" for each class and calculate the bandwidth, moreover the provider has to see whether the "X" is sending traffic which is exceeding the level so that he can make tariff accordingly. This problem is solved by the above models which can measures the packet for each class of traffic and tabulates the data. Later point of time this data can be pulled for evaluation.

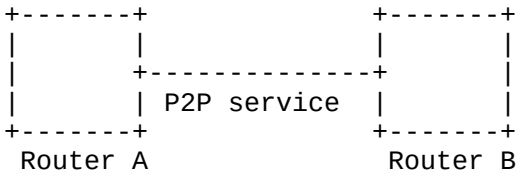


Figure 1: P2P

The same considerations can be applicable in a multipoint to multipoint scenario (e.g. VPN or Data Center interconnections). In this case Customer "X" has multiple ingress endpoints and multiple egress endpoints. The proposed matrix model is composed by the number of flows of "X" in the multipoint scenario and by class-of-service and color classification. So the SLA matrix is a reference for the analysis and evaluation phase.

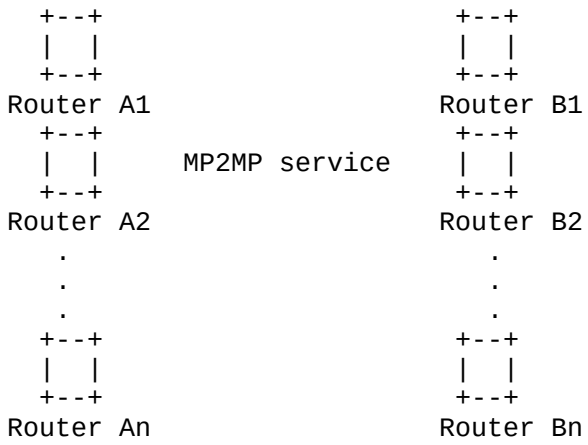


Figure 2: MP2MP

7. Acknowledgements

We would like to thank Brian Trammell for giving us the opportunity to present our draft.We would like to thank Greg Mirsky for the comments.

8. Security Considerations

This document does not introduce security issues beyond those discussed in [I.D-ietf-idr-ls-distribution] and [RFC4271].

9. IANA Considerations

IANA maintains the registry for the TLVs. BGP TE Performance TLV will require one new type code per TLV defined in this document.

10. References

10.1. Normative References

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

10.2. Informative References

[I-D.ietf-ippm-alt-mark]
Fioccola, G., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate Marking method for passive performance monitoring", draft-ietf-ippm-alt-mark-04 (work in progress), March 2017.

Authors' Addresses

Bharat M Gaonkar
Juniper Networks
1133 Innovation Way
Sunnyvale, California 94089
USA

Email: gbharat@juniper.net

Sudhin Jacob
Juniper Networks
1133 Innovation Way
Sunnyvale, California 94089
USA

Email: gbharat@juniper.net

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

Email: giuseppe.fioccola@telecomitalia.it

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

Email: bill.wu@huawei.com

Praveen Ananthasankaran
Nokia
Manyata Embassy Tech Park, Silver Oak (Wing A),
Outer Ring Road, Nagawara
Bangalore 560045
Inda

Email: praveen.ananthasankaran@nokia.com

ippm
Internet-Draft
Intended status: Experimental
Expires: November 30, 2017

F. Brockners
S. Bhandari
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JPMC
T. Mizrahi
Marvell
D. Mozes
Mellanox Technologies Ltd.
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
D. Bernier
Bell Canada

May 29, 2017

Data Fields for In-situ OAM
draft-brockners-inband-oam-data-05

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document discusses the data fields and associated data types for in-situ OAM. In-situ OAM data fields can be embedded into a variety of transports such as NSH, Segment Routing, Geneve, native IPv6 (via extension header), or IPv4. In-situ OAM can be used to complement OAM mechanisms based on e.g. ICMP or other types of probe packets.

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

This document defines data fields for "in-situ" Operations, Administration, and Maintenance (OAM). In-situ OAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than is being sent within packets specifically dedicated to OAM. A discussion of the motivation and requirements for in-situ OAM can be found in [I-D.brockners-inband-oam-requirements]. In-situ OAM is to complement mechanisms such as Ping or Traceroute, or more recent active probing mechanisms as described in [I-D.lapukhov-dataplane-probe]. In terms of "active" or "passive" OAM, "in-situ" OAM can be considered a hybrid OAM type. While no extra packets are sent, in-situ OAM adds information to the packets therefore cannot be considered passive. In terms of the classification given in [RFC7799] in-situ OAM could be portrayed as "hybrid OAM, type 1". "In-situ" mechanisms do not require extra packets to be sent and hence don’t change the packet traffic mix within the network. In-situ OAM mechanisms can be leveraged where mechanisms using e.g. ICMP do not apply or do not offer the desired results, such as proving that a certain traffic flow takes a pre-defined path, SLA verification for the live data traffic, detailed statistics on traffic distribution paths in networks that distribute traffic across multiple paths, or scenarios in which probe traffic is potentially handled differently from regular data traffic by the network devices.

2. Conventions

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].

Abbreviations used in this document:

Geneve:	Generic Network Virtualization Encapsulation [I-D.ietf-nvo3-geneve]
IOAM:	In-situ Operations, Administration, and Maintenance
MTU:	Maximum Transmit Unit
NSH:	Network Service Header [I-D.ietf-sfc-nsh]
OAM:	Operations, Administration, and Maintenance
SFC:	Service Function Chain

SID: Segment Identifier

SR: Segment Routing

VXLAN-GPE: Virtual eXtensible Local Area Network, Generic Protocol Extension [I-D.ietf-nvo3-vxlan-gpe]

3. Scope, Applicability, and Assumptions

In-situ OAM deployment assumes a set of constraints, requirements, and guiding principles which are described in this section.

Scope: This document defines the data fields and associated data types for in-situ OAM. The in-situ OAM data field can be transported by a variety of transport protocols, including NSH, Segment Routing, Geneve, IPv6, or IPv4. Encapsulation details for these different transport protocols are outside the scope of this document.

Deployment domain (or scope) of in-situ OAM deployment: IOAM is a network domain focused feature, with "network domain" being a set of network devices or entities within a single administration. For example, a network domain can include an enterprise campus using physical connections between devices or an overlay network using virtual connections / tunnels for connectivity between said devices. A network domain is defined by its perimeter or edge. Designers of carrier protocols for IOAM must specify mechanisms to ensure that in-situ OAM data stays within an IOAM domain. In addition, the operator of such a domain is expected to put provisions in place to ensure that IOAM data does not leak beyond the edge of an IOAM domain, e.g. using for example packet filtering methods. The operator should consider potential operational impact of IOAM to mechanisms such as ECMP processing (e.g. load-balancing schemes based on packet length could be impacted by the increased packet size due to IOAM), path MTU (i.e. ensure that the MTU of all links within a domain is sufficiently large to support the increased packet size due to IOAM) and ICMP message handling (i.e. in case of a native IPv6 transport, IOAM support for ICMPv6 Echo Request/Reply could desired which would translate into ICMPv6 extensions to enable IOAM data fields to be copied from an Echo Request message to an Echo Reply message).

In-situ OAM control points: IOAM data fields are added to or removed from the live user traffic by the devices which form the edge of a domain. Devices within an IOAM domain can update and/or add IOAM data-fields. Domain edge devices can be hosts or network devices.

Traffic-sets that in-situ OAM is applied to: IOAM can be deployed on all or only on subsets of the live user traffic. It SHOULD be possible to enable in-situ OAM on a selected set of traffic (e.g.,

per interface, based on an access control list or flow specification defining a specific set of traffic, etc.) The selected set of traffic can also be all traffic.

Encapsulation independence: Data formats for in-situ OAM SHOULD be defined in a transport-independent manner. In-situ OAM applies to a variety of encapsulating protocols. A definition of how IOAM data fields are carried by different transport protocols is outside the scope of this document.

Layering: If several encapsulation protocols (e.g., in case of tunneling) are stacked on top of each other, in-situ OAM data-records could be present at every layer. The behavior follows the ships-in-the-night model.

Combination with active OAM mechanisms: In-situ OAM should be usable for active network probing, enabling for example a customized version of traceroute. Decapsulating in-situ OAM nodes may have an ability to send the in-situ OAM information retrieved from the packet back to the source address of the packet or to the encapsulating node.

In-situ OAM implementation: The IOAM data-field definitions take the specifics of devices with hardware data-plane and software data-plane into account.

4. In-situ OAM Data Types and Formats

This section defines in-situ OAM data types and data fields and associated data types required for in-situ OAM. The different uses of in-situ OAM require the definition of different types of data. The in-situ OAM data fields for the data being carried corresponds to the three main categories of in-situ OAM data defined in [I-D.brockners-inband-oam-requirements], which are: edge-to-edge, per node, and for selected nodes only.

Transport options for in-situ OAM data are outside the scope of this memo, and are discussed in [I-D.brockners-inband-oam-transport]. In-situ OAM data fields are fixed length data fields. A bit field determines the set of OAM data fields embedded in a packet. Depending on the type of the encapsulation, a counter field indicates how many data fields are included in a particular packet.

In-situ OAM is expected to be deployed in a specific domain rather than on the overall Internet. The part of the network which employs in- situ OAM is referred to as the "in-situ OAM-domain". In-situ OAM data is added to a packet upon entering the in-situ OAM-domain and is removed from the packet when exiting the domain. Within the in-situ OAM-domain, the in-situ OAM data may be updated by network nodes that

the packet traverses. The device which adds an in-situ OAM data container to the packet to capture in-situ OAM data is called the "in-situ OAM encapsulating node", whereas the device which removes the in-situ OAM data container is referred to as the "in-situ OAM decapsulating node". Nodes within the domain which are aware of in-situ OAM data and read and/or write or process the in-situ OAM data are called "in-situ OAM transit nodes". Note that not every node in an in-situ OAM domain needs to be an in-situ OAM transit node. For example, a Segment Routing deployment might require the segment routing path to be verified. In that case, only the SR nodes would also be in-situ OAM transit nodes rather than all nodes.

4.1. In-situ OAM Tracing Options

"In-situ OAM tracing data" is expected to be collected at every node that a packet traverses, i.e., in a typical deployment all nodes in an in-situ OAM-domain would participate in in-situ OAM and thus be in-situ OAM transit nodes, in-situ OAM encapsulating or in-situ OAM decapsulating nodes. The maximum number of hops and the minimum path MTU of the in-situ OAM domain is assumed to be known.

To optimize hardware and software implementations tracing is defined as two separate options. Any deployment MAY choose to configure and support one or both of the following options. An implementation of the transport protocol that carries these in-situ OAM data MAY choose to support only one of the options. In the event that both options are utilized at the same time, the Incremental Trace Option MUST be placed before the Pre-allocated Trace Option.

Pre-allocated Trace Option: This trace option is defined as a container of node data fields with pre-allocated space for each node to populate its information. This option is useful for software implementations where it is efficient to allocate the space once and index into the array to populate the data during transit. The in-situ OAM encapsulating node allocates the option header and sets the fields in the option header. The in situ OAM encapsulating node allocates an array which is used to store operational data retrieved from every node while the packet traverses the domain. In-situ OAM transit nodes update the content of the array. A pointer which is part of the in-situ OAM trace data points to the next empty slot in the array, which is where the next in-situ OAM transit node fills in its data.

Incremental Trace Option: This trace option is defined as a container of node data fields where each node allocates and pushes its node data immediately following the option header. The number of node data fields recorded and maximum number of node data that can be recorded are written into the option header. This type of

trace recording is useful for some of the hardware implementations as this eliminates the need for the transit network elements to read the full array in the option and allows for arbitrarily long packets as the MTU allows. The in-situ OAM encapsulating node allocates the option header. The in-situ OAM encapsulating node based on operational state and configuration sets the fields in the header to control how large the node data list can grow. In-situ OAM transit nodes push their node data to the node data list and increment the number of node data fields in the header.

Every node data entry is to hold information for a particular in-situ OAM transit node that is traversed by a packet. The in-situ OAM decapsulating node removes the in-situ OAM data and processes and/or exports the metadata. In-situ OAM data uses its own name-space for information such as node identifier or interface identifier. This allows for a domain-specific definition and interpretation. For example: In one case an interface-id could point to a physical interface (e.g., to understand which physical interface of an aggregated link is used when receiving or transmitting a packet) whereas in another case it could refer to a logical interface (e.g., in case of tunnels).

The following in-situ OAM data is defined for in-situ OAM tracing:

- Identification of the in-situ OAM node. An in-situ OAM node identifier can match to a device identifier or a particular control point or subsystem within a device.

- Identification of the interface that a packet was received on.

- Identification of the interface that a packet was sent out on.

- Time of day when the packet was processed by the node. Different definitions of processing time are feasible and expected, though it is important that all devices of an in-situ OAM domain follow the same definition.

- Generic data: Format-free information where syntax and semantic of the information is defined by the operator in a specific deployment. For a specific deployment, all in-situ OAM nodes should interpret the generic data the same way. Examples for generic in-situ OAM data include geo-location information (location of the node at the time the packet was processed), buffer queue fill level or cache fill level at the time the packet was processed, or even a battery charge level.

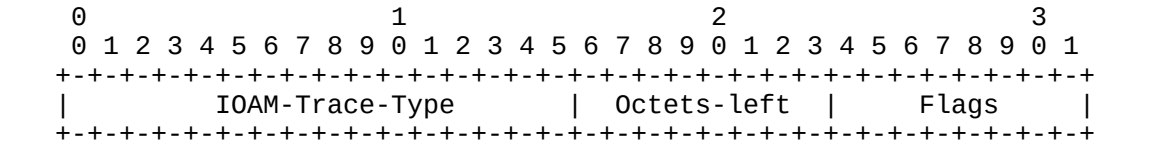
o A mechanism to detect whether in-situ OAM trace data was added at every hop or whether certain hops in the domain weren't in-situ OAM transit nodes.

The "node data list" array in the packet is populated iteratively as the packet traverses the network, starting with the last entry of the array, i.e., "node data list [n]" is the first entry to be populated, "node data list [n-1]" is the second one, etc.

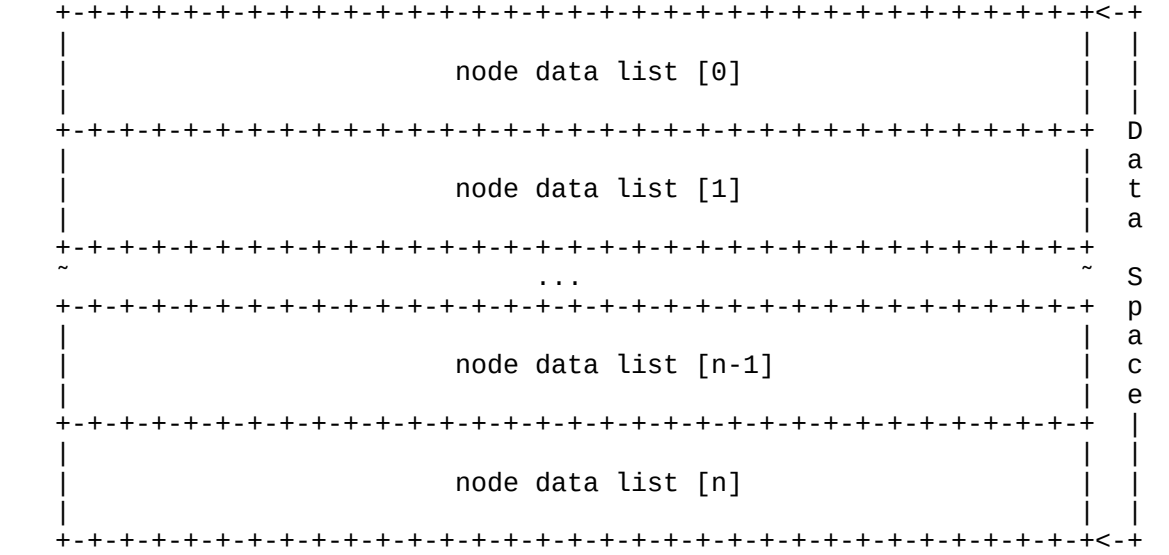
4.1.1. Pre-allocated Trace Option

In-situ OAM Pre-allocated Trace Option:

Pre-allocated Trace Option header:



Pre-allocated Trace Option Data MUST be 4-byte aligned:



IOAM-Trace-Type: A 16-bit identifier which specifies which data types are used in this node data list.

The IOAM-Trace-Type value is a bit field. The following bit fields are defined in this document, with details on each field described in the Section 4.1.3. The order of packing the data fields in each node data element follows the bit order of the IOAM-Trace-Type field, as follows:

Bit 0 (Least significant bit) When set indicates presence of Hop_Lim and node_id in the node data.

Bit 1 When set indicates presence of ingress_if_id and egress_if_id (short format) in the node data.

Bit 2 When set indicates presence of timestamp seconds in the node data

Bit 3 When set indicates presence of timestamp nanoseconds in the node data.

Bit 4 When set indicates presence of transit delay in the node data.

Bit 5 When set indicates presence of app_data (short format) in the node data.

Bit 6 When set indicates presence of queue depth in the node data.

Bit 7 When set indicates presence of variable length Opaque State Snapshot field.

Bit 8 When set indicates presence of Hop_Lim and node_id in wide format in the node data.

Bit 9 When set indicates presence of ingress_if_id and egress_if_id in wide format in the node data.

Bit 10 When set indicates presence of app_data wide in the node data.

Bit 11-15 Undefined in this draft.

Section 4.1.4 describes the in-situ OAM data types and their formats. Within an in-situ OAM domain possible combinations of these bits making the IOAM-Trace-Type can be restricted by configuration knobs.

Octets-left: 8-bit unsigned integer. It is the data space in octets remaining for recording the node data. This is used as an offset in octets in data space to record node data element.

Flags 8-bit field. The following flags are defined:

Bit 0 "Overflow" (O-bit) (most significant bit). This bit is set by the network element if there is not enough number of bytes left to record node data, no field is added and the overflow "O-bit" must be set to "1" in the header. This is useful for transit nodes to ignore further processing of the option.

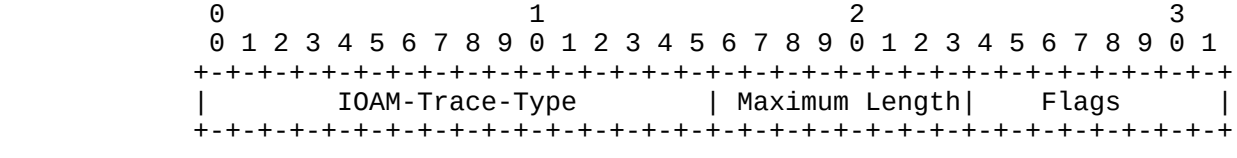
Bit 1 "Loopback" (L-bit). Loopback mode is used to send a copy of a packet back towards the source. Loopback mode assumes that a return path from transit nodes and destination nodes towards the source exists. The encapsulating node decides (e.g. using a filter) which packets loopback mode is enabled for by setting the loopback bit. The encapsulating node also needs to ensure that sufficient space is available in the IOAM header for loopback operation. The loopback bit when set indicates to the transit nodes processing this option to create a copy of the packet received and send this copy of the packet back to the source of the packet while it continues to forward the original packet towards the destination. The source address of the original packet is used as destination address in the copied packet. The address of the node performing the copy operation is used as the source address. The L-bit MUST be cleared in the copy of the packet a nodes sends it back towards the source. On its way back towards the source, the packet is processed like a regular packet with IOAM information. Once the return packet reaches the IOAM domain boundary IOAM decapsulation occurs as with any other packet containing IOAM information.

Node data List [n]: Variable-length field. The type of which is determined by the IOAM-Trace-Type representing the n-th node data in the node data list. The node data list is encoded starting from the last node data of the path. The first element of the node data list (node data list [0]) contains the last node of the path while the last node data of the node data list (node data list[n]) contains the first node data of the path traced. The index contained in "Octets-left" identifies the offset for current active node data to be populated.

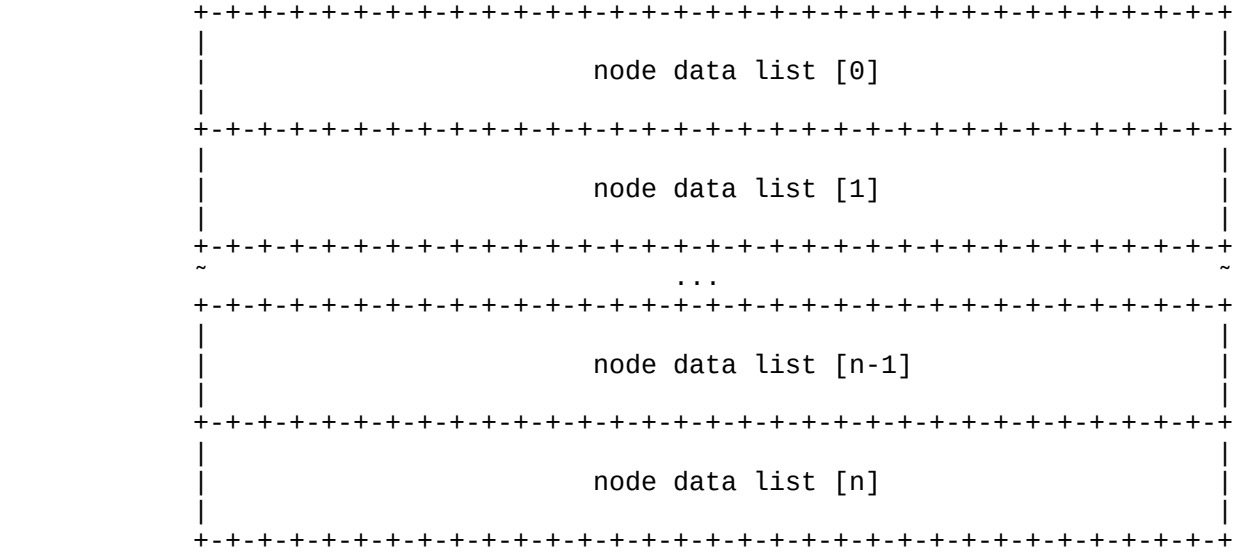
4.1.2. Incremental Trace Option

In-situ OAM Incremental Trace Option: ’

In-situ OAM Incremental Trace Option Header:



In-situ OAM Incremental Trace Option Data MUST be 4-byte aligned:



IOAM-trace-type: A 16-bit identifier which specifies which data types are used in this node data list.

The IOAM-Trace-Type value is a bit field. The following bit fields are defined in this document, with details on each field described in the Section 4.1.3. The order of packing the data fields in each node data element follows the bit order of the IOAM-Trace-Type field, as follows:

Bit 0 When set indicates presence of Hop_Lim and node_id in the node data.

Bit 1 When set indicates presence of ingress_if_id and egress_if_id in the node data.

Bit 2 When set indicates presence of timestamp seconds in the node data.

Bit 3 When set indicates presence of timestamp nanoseconds in the node data.

Bit 4 When set indicates presence of transit delay in the node data.

Bit 5 When set indicates presence of app_data in the node data.

Bit 6 When set indicates presence of queue depth in the node data.

Bit 7 When set indicates presence of variable length Opaque State Snapshot field.

Bit 8 When set indicates presence of Hop_Lim and node_id wide in the node data.

Bit 9 When set indicates presence of ingress_if_id and egress_if_id wide in the node data.

Bit 10 When set indicates presence of app_data wide in the node data.

Bit 11-15 Undefined in this draft.

Section 4.1.4 describes the in-situ OAM data types and their formats.

Maximum Length: 8-bit unsigned integer. This field specifies the maximum length of the node data list in octets. Given that the sender knows the minimum path MTU, the sender can set the maximum of node data bytes allowed before exceeding the MTU. Thus, a simple comparison between "Opt data Len" and "Max Length" allows to decide whether or not data could be added.

Flags 8-bit field. Following flags are defined:

Bit 0 "Overflow" (O-bit) (least significant bit). This bit is set by the network element if there is not enough number of bytes left to record node data, no field is added and the overflow "O-bit" must be set to "1" in the header. This is

useful for transit nodes to ignore further processing of the option.

Bit 1 "Loopback" (L-bit). This bit when set indicates to the transit nodes processing this option to send a copy of the packet back to the source of the packet while it continues to forward the original packet towards the destination. The L-bit MUST be cleared in the copy of the packet before sending it.

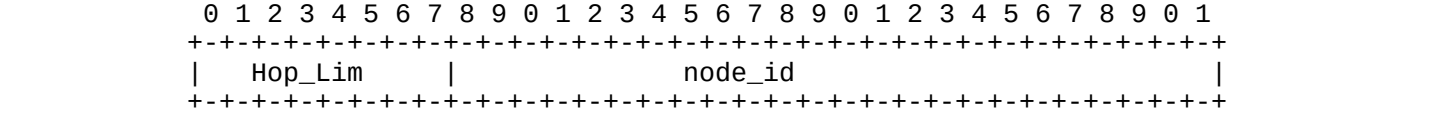
Node data List [n]: Variable-length field. The type of which is determined by the OAM Type representing the n-th node data in the node data list. The node data list is encoded starting from the last node data of the path. The first element of the node data list (node data list [0]) contains the last node of the path while the last node data of the node data list (node data list[n]) contains the first node data of the path traced.

4.1.3. In-situ OAM node data fields and associated formats

All the data fields MUST be 4-byte aligned. The IOAM encapsulating node MUST initialize data fields that it adds to the packet to zero. If a node which is supposed to update an IOAM data field is not capable of populating the value of a field set in the IOAM-Trace-Type, the field value MUST be left unaltered except when explicitly specified in the field description below. In the description of data below if zero is valid value then a non-zero value to mean not populated is specified.

Data field and associated data type for each of the data field is shown below:

Hop_Lim and node_id: 4-octet field defined as follows:

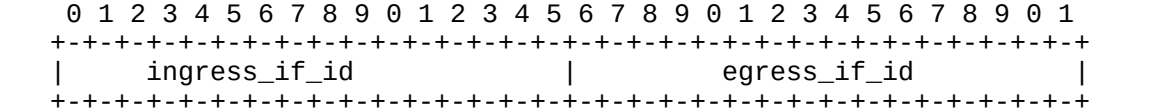


Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop Limit information is used to identify the location of the node in the communication path. This is copied from the lower layer, e.g., TTL value in IPv4 header or hop limit field from IPv6 header of the packet when the packet is ready for transmission.

node_id: 3-octet unsigned integer. Node identifier field to uniquely identify a node within in-situ OAM domain. The

procedure to allocate, manage and map the node_ids is beyond the scope of this document.

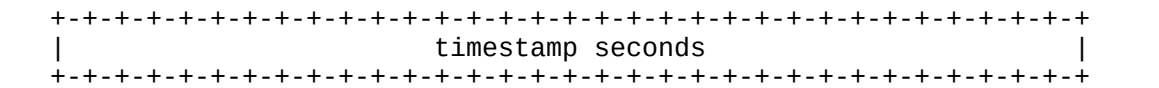
ingress_if_id and egress_if_id: 4-octet field defined as follows: When this field is part of the data field but a node populating the field is not able to fill it, the position in the field must be filled with value 0xFFFFFFFF to mean not populated.



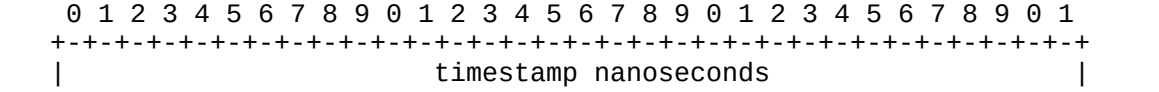
ingress_if_id: 2-octet unsigned integer. Interface identifier to record the ingress interface the packet was received on.

egress_if_id: 2-octet unsigned integer. Interface identifier to record the egress interface the packet is forwarded out of.

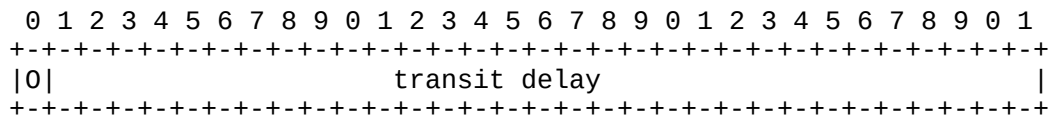
timestamp seconds: 4-octet unsigned integer. Absolute timestamp in seconds that specifies the time at which the packet was received by the node. The structure of this field is identical to the most significant 32 bits of the 64 least significant bits of the [IEEE1588v2] timestamp. This truncated field consists of a 32-bit seconds field. As defined in [IEEE1588v2], the timestamp specifies the number of seconds elapsed since 1 January 1970 00:00:00 according to the International Atomic Time (TAI).



timestamp nanoseconds: 4-octet unsigned integer in the range 0 to 10^9-1. This timestamp specifies the fractional part of the wall clock time at which the packet was received by the node in units of nanoseconds. This field is identical to the 32 least significant bits of the [IEEE1588v2] timestamp. This fields allows for delay computation between any two nodes in the network when the nodes are time synchronized. When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.

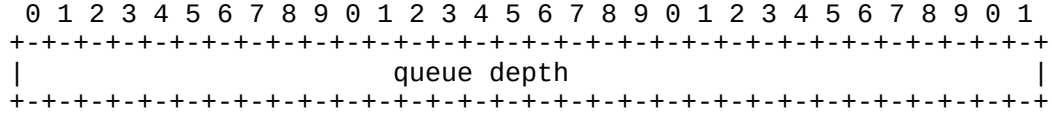


transit delay: 4-octet unsigned integer in the range 0 to 2^30-1. It is the time in nanoseconds the packet spent in the transit node. This can serve as an indication of the queuing delay at the node. If the transit delay exceeds 2^30-1 nanoseconds then the top bit 'O' is set to indicate overflow. When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.

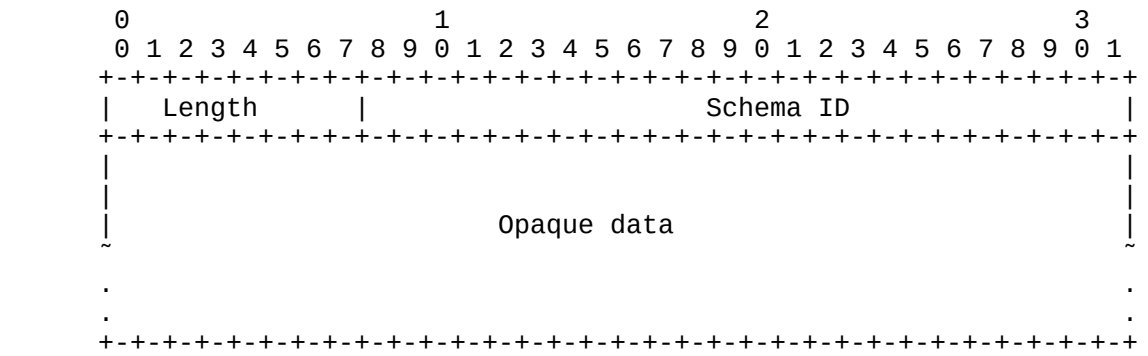


app_data: 4-octet placeholder which can be used by the node to add application specific data

queue depth: 4-octet unsigned integer field. This field indicates the current length of the egress interface queue of the interface from where the packet is forwarded out. The queue depth is expressed as the current number of memory buffers used by the queue (a packet may consume one or more memory buffers, depending on its size). When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.



Opaque State Snapshot: Variable length field. It allows the network element to store an arbitrary state in the node data field , without a pre-defined schema. The schema needs to be made known to the analyzer by some out-of-band mechanism. The specification of this mechanism is beyond the scope of this document. The 24-bit "Schema Id" field in the field indicates which particular schema is used, and should be configured on the network element by the operator.

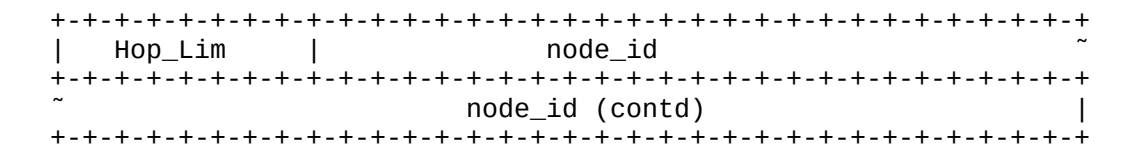


Length: 1-octet unsigned integer. It is the length in octets of the Opaque data field that follows Schema Id. It MUST always be a multiple of 4.

Schema ID: 3-octet unsigned integer identifying the schema of Opaque data.

Opaque data: Variable length field. This field is interpreted as specified by the schema identified by the Schema ID.

Hop_Lim and node_id wide: 8-octet field defined as follows:

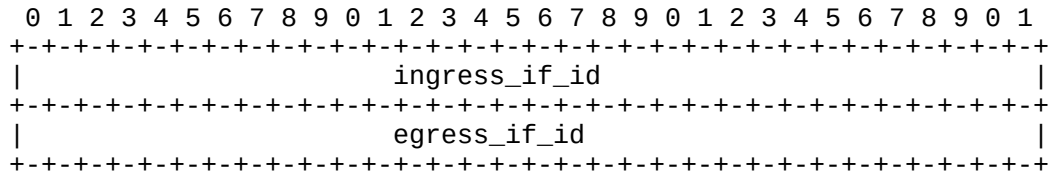


Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop Limit information is used to identify the location of the node in the communication path. This is copied from the lower layer for e.g. TTL value in IPv4 header or hop limit field from IPv6 header of the packet.

node_id: 7-octet unsigned integer. Node identifier field to uniquely identify a node within in-situ OAM domain. The procedure to allocate, manage and map the node_ids is beyond the scope of this document.

ingress_if_id and egress_if_id wide: 8-octet field defined as follows: When this field is part of the data field but a node populating the field is not able to fill it, the field position in

the field must be filled with value 0xFFFFFFFF to mean not populated.



ingress_if_id: 4-octet unsigned integer. Interface identifier to record the ingress interface the packet was received on.

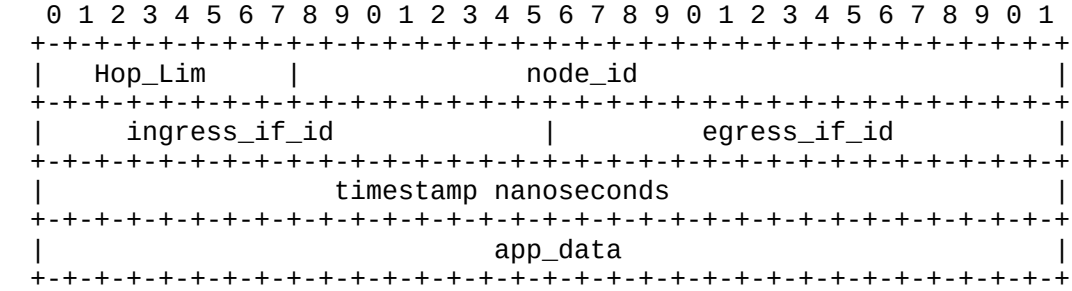
egress_if_id: 4-octet unsigned integer. Interface identifier to record the egress interface the packet is forwarded out of.

app_data wide: 8-octet placeholder which can be used by the node to add application specific data.

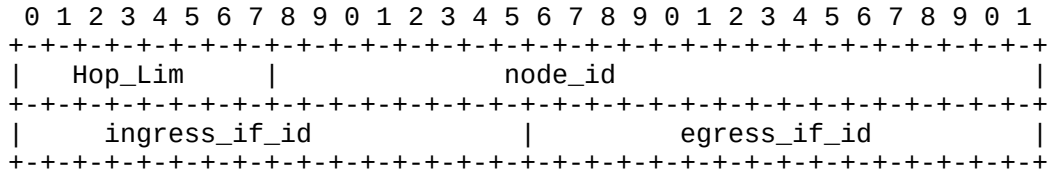
4.1.4. Examples of In-situ OAM node data

An entry in the "node data list" array can have different formats, following the needs of the deployment. Some deployments might only be interested in recording the node identifiers, whereas others might be interested in recording node identifier and timestamp. The section defines different types that an entry in "node data list" can take.

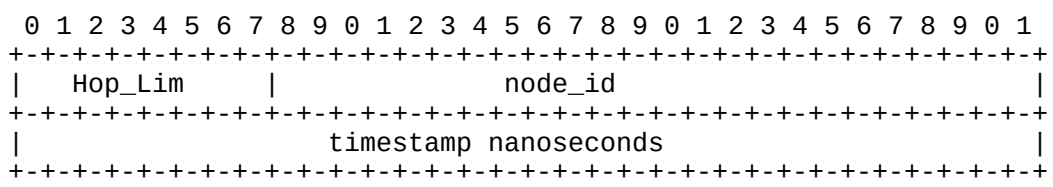
0x002B: IOAM-Trace-Type is 0x2B then the format of node data is:



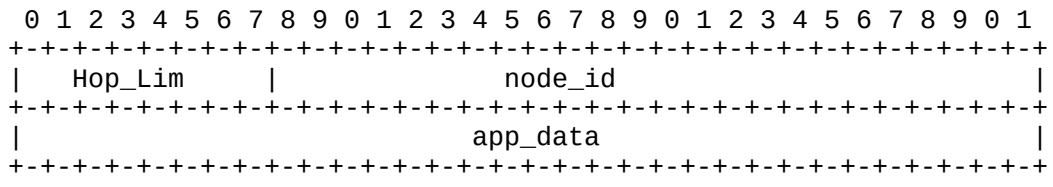
0x0003: IOAM-Trace-Type is 0x0003 then the format is:



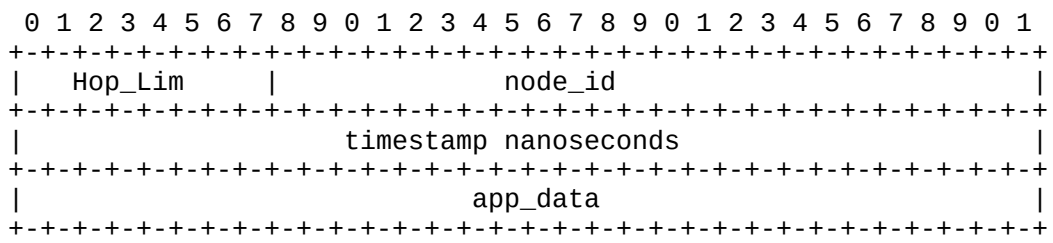
0x0009: IOAM-Trace-Type is 0x0009 then the format is:



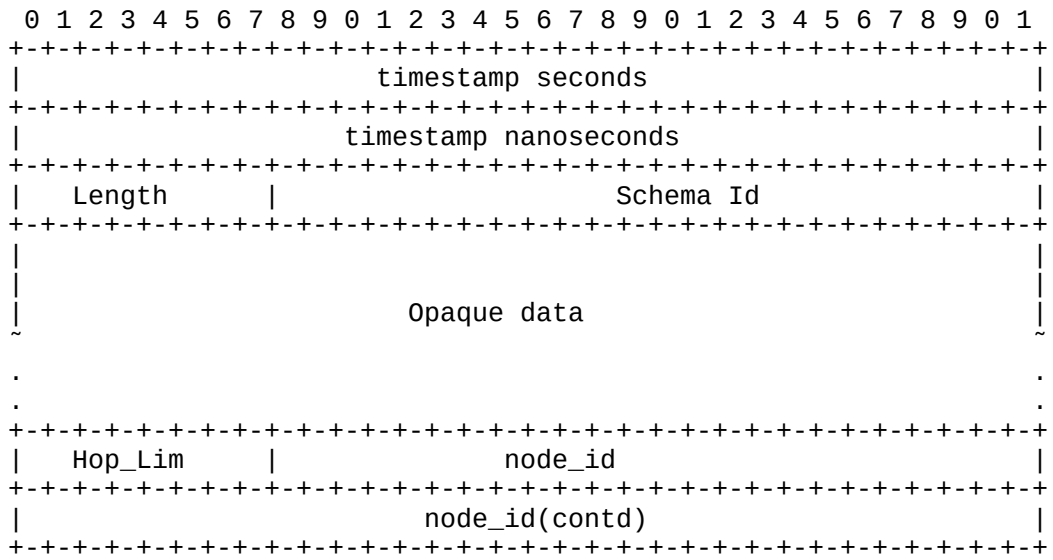
0x0021: IOAM-Trace-Type is 0x0021 then the format is:



0x0029: IOAM-Trace-Type is 0x0029 then the format is:



0x018C: IOAM-Trace-Type is 0x104D then the format is:



4.2. In-situ OAM Proof of Transit Option

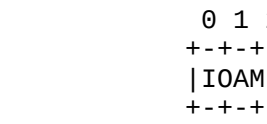
In-situ OAM Proof of Transit data is to support the path or service function chain [RFC7665] verification use cases. Proof-of-transit uses methods like nested hashing or nested encryption of the in-situ OAM data or mechanisms such as Shamir's Secret Sharing Schema (SSSS). While details on how the in-situ OAM data for the proof of transit option is processed at in-situ OAM encapsulating, decapsulating and transit nodes are outside the scope of the document, all of these approaches share the need to uniquely identify a packet as well as iteratively operate on a set of information that is handed from node to node. Correspondingly, two pieces of information are added as in-situ OAM data to the packet:

- Random: Unique identifier for the packet (e.g., 64-bits allow for the unique identification of 2^64 packets).

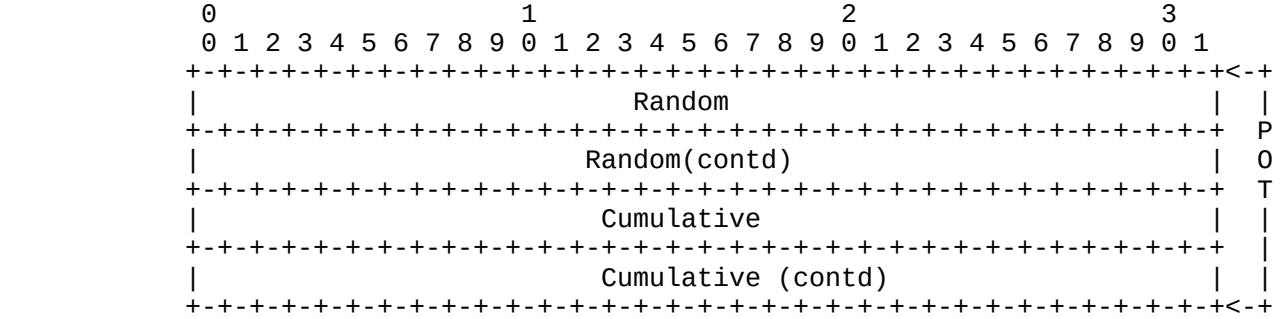
- Cumulative: Information which is handed from node to node and updated by every node according to a verification algorithm.

In-situ OAM Proof of Transit Option:

In-situ OAM Proof of Transit Option Header:



In-situ OAM Proof of Transit Option Data MUST be 4-byte aligned:



IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included. This document defines POT Type 0:

0: POT data is a 16 Octet field as described below.

Profile to use (P): 1-bit. Indicates which POT-profile is used to generate the Cumulative. Any node participating in POT will have a maximum of 2 profiles configured that drive the computation of cumulative. The two profiles are numbered 0, 1. This bit conveys whether profile 0 or profile 1 is used to compute the Cumulative.

Random: 64-bit Per packet Random number.

Cumulative: 64-bit Cumulative that is updated at specific nodes by processing per packet Random number field and configured parameters.

Note: Larger or smaller sizes of "Random" and "Cumulative" data are feasible and could be required for certain deployments (e.g. in case of space constraints in the transport protocol used). Future versions of this document will address different sizes of data for "proof of transit".

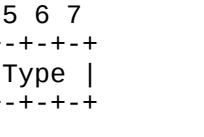
4.3. In-situ OAM Edge-to-Edge Option

The in-situ OAM Edge-to-Edge Option is to carry data that is added by the in-situ OAM encapsulating node and interpreted by in-situ OAM decapsulating node. The in-situ OAM transit nodes MAY process the data without modifying it.

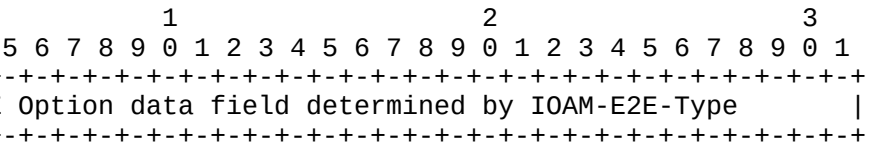
Currently only sequence numbers use the in-situ OAM Edge-to-Edge option. In order to detect packet loss, packet reordering, or packet duplication in an in-situ OAM-domain, sequence numbers can be added to packets of a particular tube (see [I-D.hildebrand-spud-prototype]). Each tube leverages a dedicated namespace for its sequence numbers.

In-situ OAM Edge-to-Edge Option:

In-situ OAM Edge-to-Edge Option Header:



In-situ OAM Edge-to-Edge Option Data MUST be 4-byte aligned:



IOAM-E2E-Type: 8-bit identifier of a particular in situ OAM E2E variant.

0: E2E option data is a 64-bit sequence number added to a specific tube which is used to identify packet loss and reordering for that tube.

5. In-situ OAM Data Export

In-situ OAM nodes collect information for packets traversing a domain that supports in-situ OAM. The device at the domain edge (which could also be an end-host) which receives a packet with in-situ OAM information chooses how to process the in-situ OAM data collected within the packet. This decapsulating node can simply discard the information collected, can process the information further, or export the information using e.g., IPFIX.

The discussion of in-situ OAM data processing and export is left for a future version of this document.

6. IANA Considerations

IANA considerations will be added in a future version of this document.

7. Manageability Considerations

Manageability considerations will be addressed in a later version of this document..

8. Security Considerations

Security considerations will be addressed in a later version of this document. For a discussion of security requirements of in-situ OAM, please refer to [I-D.brockners-inband-oam-requirements].

9. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, LJ Wobker, Erik Nordmark, Vengada Prasad Govindan, and Andrew Yourtchenko for the comments and advice. This document leverages and builds on top of several concepts described in [I-D.kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

10. References

10.1. Normative References

[I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[IEEE1588v2]
Institute of Electrical and Electronics Engineers, "1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588-2008, 2008, <http://standards.ieee.org/findstds/standard/1588-2008.html>.

[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>.

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

10.2. Informative References

[I-D.brockners-inband-oam-transport]
Brockners, F., Bhandari, S., Govindan, V., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., and R. <>, "Encapsulations for In-situ OAM Data", draft-brockners-inband-oam-transport-03 (work in progress), March 2017.

[I-D.hildebrand-spud-prototype]
Hildebrand, J. and B. Trammell, "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in progress), March 2015.

[I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-04 (work in progress), March 2017.

[I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 2017.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-12 (work in progress), February 2017.

[I-D.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.

[I-D.lapukhov-dataplane-probe]
Lapukhov, P. and r. remy@barefootnetworks.com, "Data-plane probe for in-band telemetry collection", draft-lapukhov-dataplane-probe-01 (work in progress), June 2016.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <http://www.rfc-editor.org/info/rfc7665>.

Authors' Addresses

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

Shwetha Bhandari
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Email: shwethab@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709
United States

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

Stephen Youell
JP Morgan Chase
25 Bank Street
London E14 5JP
United Kingdom

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 2066721
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Petr Lapukhov
Facebook
1 Hacker Way
Menlo Park, CA 94025
US

Email: petr@fb.com

Remy Chang
Barefoot Networks
2185 Park Boulevard
Palo Alto, CA 94306
US

Daniel
Bell Canada

Email: daniel.bernier@bell.ca

Network Working Group
Internet-Draft
Updates: 2330 (if approved)
Intended status: Informational
Expires: September 7, 2017

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
March 6, 2017

IPv6 Updates for IPPM's Active Metric Framework
draft-ietf-ippm-2330-ipv6-01

Abstract

This memo updates the IP Performance Metrics (IPPM) Framework RFC 2330 with new considerations for measurement methodology and testing. It updates the definition of standard-formed packets in RFC 2330 to include IPv6 packets and augments distinguishing aspects of packets, referred to as Type-P for test packets in RFC 2330. This memo identifies that IPv4-IPv6 co-existence can challenge measurements within the scope of the IPPM Framework. Exemplary use cases include, but are not limited to IPv4-IPv6 translation, NAT, protocol encapsulation, or IPv6 header compression.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 7, 2017.

Table of Contents

1. Introduction

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

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

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

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

2. Scope

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

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

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

3. Packets of Type-P

A fundamental property of many Internet metrics is that the measured value of the metric depends on characteristics of the IP packet(s) used to make the measurement. Potential influencing factors include IP header fields and their values, but also higher-layer protocol headers and their values. Consider an IP-connectivity metric: one obtains different results depending on whether one is interested in connectivity for packets destined for well-known TCP ports or unreserved UDP ports, or those with invalid IPv4 checksums, or those with TTL or Hop Limit of 16, for example. In some circumstances these distinctions will result in special treatment of packets in intermediate nodes and end systems (for example, if Diffserv [RFC2780], ECN [RFC3168], Router Alert, Hop-by-hop extensions [RFC7045], or Flow Labels [RFC6437] are used, or in the presence of firewalls or RSVP reservations).

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

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

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

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

A closely related note: it would be very useful to know if a given Internet component (like host, link, or path) treats equally a class C of different types of packets. If so, then any one of those types of packets can be used for subsequent measurement of the component. This suggests we devise a metric or suite of metrics that attempt to determine C.

Load balancing over parallel paths is one particular example where such a class C would be more complex to determine in IPPM measurements. Load balancers often use flow identifiers, computed as hashes of (specific parts of) the packet header, for deciding among the available parallel paths a packet will traverse. Packets with identical hashes are assigned to the same flow and forwarded to the same resource in the load balancer's pool. The presence of a load balancer on the measurement path, as well as the specific headers and fields that are used for the forwarding decision, are not known when measuring the path as a black-box. Potential assessment scenarios

include the measurement of one of the parallel paths, and the measurement of all available parallel paths that the load balancer can use. Knowledge of a load balancer's flow definition (alternatively: its class C specific treatment in terms of header fields in scope of hash operations) is therefore a prerequisite for repeatable measurements. A path may have more than one stage of load balancing, adding to class C definition complexity.

4. Standard-Formed Packets

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

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

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

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

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

- All parameters used in the header and Extension Headers are found in the IANA Registry of Internet Protocol Version 6 (IPv6) Parameters, partly specified in [RFC7045].

Compressed IPv6 headers must be compliant with [RFC4494], as updated by [RFC6282], in order to be declared "standard-formed".

The topic of IPv6 Extension Headers brings current controversies into focus as noted by [RFC6564] and [RFC7045]. The following additional considerations apply when IPv6 Extension Headers are present:

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

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

- Extension Header insertion or deletion: It is possible that Extension Headers could be added to, or removed from the header chain. The resulting packet may be standard-formed, with a corresponding Type-P.

- A change in packet length (from the corresponding packet observed at the Source) or header modification is a significant factor in Internet measurement, and requires a new Type-P to be reported.

We further require that if a packet is described as having a "length of B octets", then 0 <= B <= 65535; and if B is the payload length in octets, then B <= (65535-IP header size in octets, including any Extension Headers). The jumbograms defined in [RFC2675] are not covered by this length analysis. In practice, the path MTU will restrict the length of standard-formed packets that can successfully traverse the path.

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

A particular type of standard-formed packet often useful to consider is the "minimal IP packet from A to B" - this is an IP packet with the following properties:

+ It is standard-formed.

+ Its data payload is 0 octets.

+ It contains no options or Extension Headers.

(Note that we do not define its protocol field, as different values may lead to different treatment by the network.)

When defining IP metrics we keep in mind that no packet smaller or simpler than this can be transmitted over a correctly operating IP network.

5. NAT, IPv4-IPv6 Transition and Compression Techniques

This memo adds the key considerations for utilizing IPv6 in two critical conventions of the IPPM Framework, namely packets of Type-P and standard-formed packets. The need for co-existence of IPv4 and IPv6 has originated transitioning standards like the Framework for IPv4/IPv6 Translation in [RFC6144] or IP/ICMP Translation Algorithms in [RFC6145] and [RFC7757].

The definition and execution of measurements within the context of the IPPM Framework is challenged whenever such translation mechanisms are present along the measurement path. In particular use cases like IPv4-IPv6 translation, NAT, protocol encapsulation, or IPv6 header compression may result in modification of the measurement packet’s

Type-P along the path. All these changes must be reported. Exemplary consequences include, but are not limited to:

- Modification or addition of headers or header field values in intermediate nodes. As noted in Section 4 for IPv6 extension header manipulation, NAT, IPv4-IPv6 transitioning or IPv6 header compression mechanisms may result in changes of the measurement packets' Type-P, too. Consequently, hosts along the measurement path may treat packets differently because of the Type-P modification. Measurements at observation points along the path may also need extra context to uniquely identify a packet.

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

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

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

Points that are worthwhile discussing further: handling of large packets in IPv6 (including fragment extension headers, PMTUD, PLMTUD), extent of coverage for 6LO and IPv6 Header Compression, and the continued need to define a "minimal standard-formed packet".

.

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>.

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

[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>.

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

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, DOI 10.17487/RFC6145, April 2011, <http://www.rfc-editor.org/info/rfc6145>.

[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>.

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

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: Experimental
Expires: September 2, 2017

G. Fioccola, Ed.
A. Capello, Ed.
M. Cociglio
L. Castaldelli
Telecom Italia
M. Chen, Ed.
L. Zheng, Ed.
Huawei Technologies
G. Mirsky, Ed.
ZTE
T. Mizrahi, Ed.
Marvell

March 1, 2017

Alternate Marking method for passive performance monitoring
draft-ietf-ippm-alt-mark-04

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.

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 2, 2017.

Table of Contents

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

The lack of adequate tools to measure packet loss with the desired accuracy drove an effort to design a new method for the performance monitoring of live traffic, 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.

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.

This document is organized as follows:

- Section 2 gives an overview of the method, including a comparison with different measurement strategies;
- Section 3 describes the method in detail;
- Section 4 reports considerations about synchronization, data correlation and packet re-ordering;
- Section 5 reports examples of implementation and deployment of the method. Furthermore the operational experiment done at Telecom Italia is described;
- Section 8 includes some security aspects;
- Section 9 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.

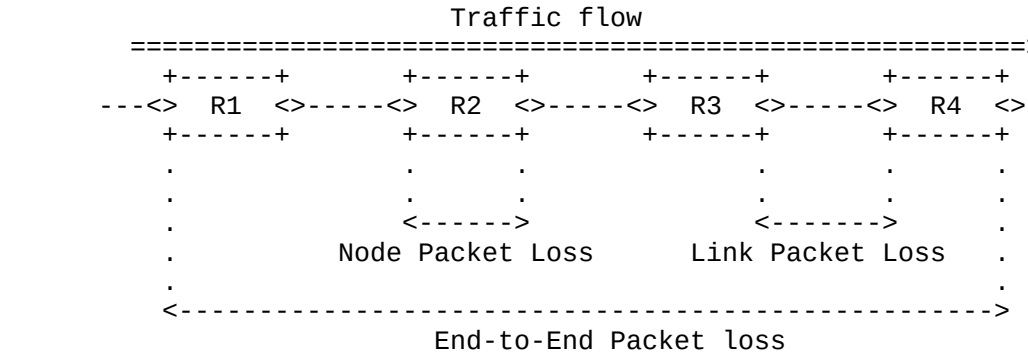


Figure 1: Available measurements

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

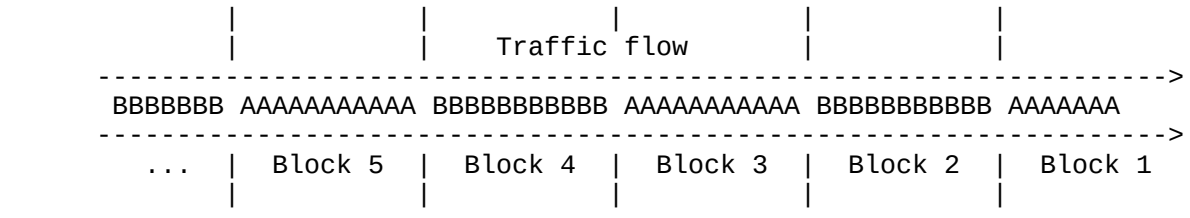


Figure 2: Traffic coloring

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.

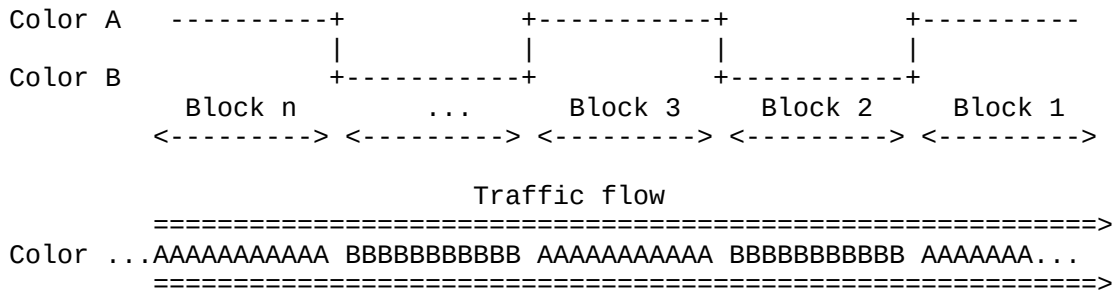


Figure 3: Computation of link packet loss

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.

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.

3.1.1. Timing aspects

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

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

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

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

The following figure eplains both issues.

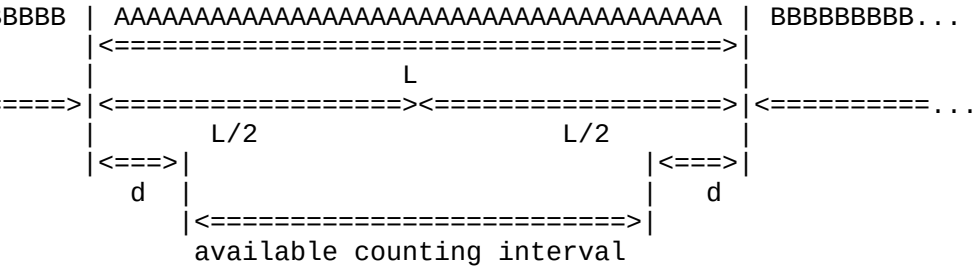


Figure 4: Timing aspects

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

The guardband d is given by:

d = A + D_max - D_min,

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

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

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

d < L/2.

3.2. 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 2, R1 stores a timestamp TS(A1)R1 when it sends the first packet of block 1 (A-colored), a timestamp TS(B2)R1 when it sends the first packet of block 2 (B-colored) and so on for every other block. R2 performs the same operation on the

receiving side, recording TS(A1)R2, TS(B2)R2 and so on. Since the timestamps refer to specific packets (the first packet of each block) we are sure that timestamps compared to compute delay refer to the same packets. By comparing TS(A1)R1 with TS(A1)R2 (and similarly TS(B2)R1 with TS(B2)R2 and so on) it is possible to measure the delay between R1 and R2. In order to have more measurements, it is possible to take and store more timestamps, referring to other packets within each block.

In order to coherently compare timestamps collected on different routers, the 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).

3.2.1.1. Mean delay

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

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

3.2.2. Double marking methodology

The Single marking methodology for one-way delay measurement is sensitive to out of order reception of packets. The first approach to overcome this problem is described before and is based on the concept of mean delay. But the limitation of mean delay is that it

doesn’t give information about the delay values distribution for the duration of the block. Additionally it may be useful to have not only the mean delay but also the minimum, maximum and median delay values and, in wider terms, to know more about the statistic distribution of delay values. So in order to have more information about the delay and to overcome out of order issues, a different approach can be introduced: it is based on double marking methodology.

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

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

3.3. 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. We refer to the definition in RFC 3393 [RFC3393]. The alternation of colors, for single marking method, can be used as a time reference to measure delay variations. In case of double marking, the time reference is given by the second marked packets. Considering the example depicted in Figure 2, R1 stores a timestamp TS(A)R1 whenever it sends the first packet of a block and R2 stores a timestamp TS(B)R2 whenever it receives the first packet of a block. The inter-arrival jitter can be easily derived from one-way delay measurement, by evaluating the delay variation of consecutive samples.

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

4. Considerations

This section highlights some considerations about the methodology.

4.1. Synchronization

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

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

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

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

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

4.2. Data Correlation

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

- o A possibility is to use a centralized solution using Network Management System (NMS) to correlate data;
- o Another possibility is to define a protocol based distributed solution, by defining a new protocol or by extending the existing protocols (e.g. RFC6374, TWAMP, OWAMP) in order to communicate the counters and timestamps between nodes.

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

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

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

When the nodes or NMS see, for example, same BNs associated with two packet counts from an upstream and a downstream node respectively, it considers that these two packet counts corresponding to the same block, i.e. that these two packet counts belong to the same block of markers from the upstream and downstream node. The assumption of this BN mechanism is that the measurement nodes are time synchronized. This requires the measurement nodes to have a certain

time synchronization capability (e.g., the Network Time Protocol (NTP) [RFC5905], or the IEEE 1588 Precision Time Protocol (PTP) [IEEE1588]). Synchronization aspects are further discussed in Section 4.

4.3. Packet Re-ordering

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

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

Figure 5: Packet Reordering

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

Most of the packet re-ordering occur at the edge of adjacent blocks, and they are easy to handle if the interval of each block is sufficient large. Then, it can assume that the packets with different marker belong to the block that they are more close to. If the interval is small, it is difficult and sometime impossible to determine to which block a packet belongs. See above example, the packet with the marker of "B" in block 3, there is no safe way to tell whether the packet belongs to block 2 or block 4.

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

5. 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.

5.1. Report on the operational experiment at Telecom Italia

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 methodology has been applied 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:

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

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

- 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.

5.1.1. 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 ca be used 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).

5.1.2. 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 od 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.

5.1.3. 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.

5.1.4. 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 anaware that packets were marked. So this restoring technique makes Alternate Marking completely transparent outside its monitoring domain.

5.2. IP flow performance measurement (IPFPM)

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

5.3. 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.ietf-bier-pmmm-oam] details the measurement for multicast service over BIER domain.

5.4. Overlay OAM Passive Performance Measurement

The Overlay OAM Design Team is considering the preliminary OAM requirements from NVO3, BIER, and SFC. Marking Method is the preferred passive method to measure performance.

[I-D.ooamdt-rtgwg-ooam-requirement] and
[I-D.ooamdt-rtgwg-oam-gap-analysis] explain in deep this item.

5.5. 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.

5.6. Application to active performance measurement

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

6. Hybrid measurement

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

7. Compliance with RFC6390 guidelines

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

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

used a subset of the Performance Metric Definition template defined by [RFC6390].

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

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

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

- Dependencies: the values of the counters have to be correlated to the time interval they refer to; moreover, as far the 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.

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

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

8. Security Considerations

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

There are two types of security concerns: potential harm caused by the measurements and potential harm to the measurements. 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 marking of the packets, or by an attacker injecting artificial traffic. Authentication techniques, such as digital signatures, may be used where appropriate to guard against injected traffic attacks.

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.

The measurement itself may be affected by routers (or other network devices) along the path of IP packets intentionally altering the value of marking bits of packets. As mentioned above, the mechanism specified in this document is just in the context of one Service Provider's network, and thus the routers (or other network devices) are locally administered and this type of attack can be avoided.

One of the main security threats in OAM protocols is network reconnaissance; an attacker can gather information about the network performance by passively eavesdropping to OAM messages. The advantage of the methods described in this document is that the marking bits are the only information that is exchanged between the network devices. Therefore, passive eavesdropping to data plane traffic does not allow attackers to gain information about the network performance.

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

9. Conclusions

The advantages of the method described in this document are:

- easy implementation: it can be implemented using features already available on major routing platforms;
- low computational effort: the additional load on processing is negligible;
- accurate packet loss measurement: single packet loss granularity is achieved with a passive measurement;
- 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 protocol extension, but it could be further improved by means of some extension to existing protocols. Specifically, the use of DiffServ bits for coloring the packets could not be a viable solution in some cases: a standard method to color the packets for this specific application could be beneficial.

10. IANA Considerations

There are no IANA actions required.

11. Acknowledgements

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.ietf-bier-pmmm-oam] [I-D.chen-ippm-coloring-based-ipfpm-framework]).

In addition 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.

12. References

12.1. 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>.

12.2. Informative References

[I-D.bryant-mpls-rfc6374-sfl]
Bryant, S., Chen, M., Li, Z., Swallow, G., Sivabalan, S., Mirsky, G., and G. Fioccola, "RFC6374 Synonymous Flow Labels", draft-bryant-mpls-rfc6374-sfl-03 (work in progress), October 2016.

[I-D.bryant-mpls-sfl-framework]
Bryant, S., Chen, M., Li, Z., Swallow, G., Sivabalan, S., and G. Mirsky, "Synonymous Flow Label Framework", draft-bryant-mpls-sfl-framework-02 (work in progress), October 2016.

[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-alt-mark-active]
Fioccola, G., Clemm, A., Cociglio, M., Chandramouli, M., and A. Capello, "Alternate Marking Extension to Active Measurement Protocol", draft-fioccola-ippm-alt-mark-active-00 (work in progress), July 2016.

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

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication in MPLS and non-MPLS Networks", draft-ietf-bier-mpls-encapsulation-06 (work in progress), December 2016.

[I-D.ietf-bier-pmmm-oam]
Mirsky, G., Zheng, L., Chen, M., and G. Fioccola, "Performance Measurement (PM) with Marking Method in Bit Index Explicit Replication (BIER) Layer", draft-ietf-bier-pmmm-oam-01 (work in progress), January 2017.

[I-D.ietf-mpls-flow-ident]
Bryant, S., Pignataro, C., Chen, M., Li, Z., and G. Mirsky, "MPLS Flow Identification Considerations", draft-ietf-mpls-flow-ident-04 (work in progress), February 2017.

[I-D.ooamdt-rtgwg-oam-gap-analysis]
Mirsky, G., Nordmark, E., Pignataro, C., Kumar, N., Kumar, D., Chen, M., Yizhou, L., Mozes, D., Networks, J., and I. Bagdonas, "Operations, Administration and Maintenance (OAM) for Overlay Networks: Gap Analysis", draft-ooamdt-rtgwg-oam-gap-analysis-02 (work in progress), July 2016.

[I-D.ooamdt-rtgwg-ooam-requirement]
Kumar, N., Pignataro, C., Kumar, D., Mirsky, G., Chen, M., Nordmark, E., Networks, J., and D. Mozes, "Overlay OAM Requirements", draft-ooamdt-rtgwg-ooam-requirement-02 (work in progress), January 2017.

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

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

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <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>.

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

Authors' Addresses

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

Email: giuseppe.fioccola@telecomitalia.it

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

Email: alessandro.capello@telecomitalia.it

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

Email: mauro.cociglio@telecomitalia.it

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

Email: luca.castaldelli@telecomitalia.it

Mach(Guoyi) Chen (editor)
Huawei Technologies

Email: mach.chen@huawei.com

Lianshu Zheng (editor)
Huawei Technologies

Email: vero.zheng@huawei.com

Greg Mirsky (editor)
ZTE
USA

Email: gregimirsky@gmail.com

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

Email: talmi@marvell.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2017

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

March 9, 2017

Initial Performance Metric Registry Entries
draft-ietf-ippm-initial-registry-03

Abstract

This memo defines the Initial Entries for the Performance Metrics Registry. This version includes:

* All section 4, 5, 6, 7, and 8 parameters reference YANG types for alternate data formats.

* implementation of standard naming format for parameters.

* implementation of many IANA early-review comments.

Still need: Add MBM metric entry.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2017.

Table of Contents

1. Introduction

Note: Efforts to synchronize structure and terminology with [I-D.ietf-ippm-metric-registry] will likely be incomplete until both drafts are stable.

This memo proposes an initial set of entries for the Performance Metric Registry. It uses terms and definitions from the IPPM literature, primarily [RFC2330]. Proponents of Passive Performance Metrics are encouraged to develop a similar document.

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

IETF-wide registry of metrics. While examinating aspects of metric specifications which need to be registered, it became clear that none of the existing metric templates fully satisfies the particular needs of a registry.

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

2. Scope

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

3. Registry Categories and Columns

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

Registry Categories and Columns, shown as

	Category	
	Column	Column

Summary

Identifier	Name	URIs	Desc.	Reference	Change Controller	Ver

Metric Definition

Reference Definition	Fixed Parameters

Method of Measurement

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

Output

Type	Reference Definition	Units	Calibration

Administrative Information

Status	Request	Rev	Rev.Date

Comments and Remarks

4. UDP Round-trip Latency 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 a single section can specify 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>

RTDelay_Active_IP-UDP-Poisson_RFCXXXXsecY_Seconds_95Percentile

4.1.3. URIs

URN: Prefix urn:ietf:metrics:perf:<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.

4.1.5. Change Controller

IETF

4.1.6. Version (of Registry Format)

1.0

4.2. Metric Definition

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

4.2.1. Reference Definition

<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-singleton sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

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

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

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 as defined in Section 13 of [RFC2330]:

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

* total of 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 Stream Generation

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

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

4.4.2. Reference Definition

<describe the reference 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 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].

Raw -- REMOVED IN VERSION 01

For Act_IP_UDP_Round-trip_Delay_Poisson_95th-percentile:

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

4.4.3. Metric Units

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

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

4.4.4. Calibration

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

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

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

4.5. Administrative items

4.5.1. Status

<current 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>

OWPDV_Active_IP-UDP-Poisson_RFCXXXXsecY_Seconds_95Percentile

5.1.3. URIs

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

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

5.1.4. Description

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

5.1.5. Change Controller

<org or person >

IETF

5.1.6. Version (of Registry Format)

1.0

5.2. Metric Definition

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

5.2.1. Reference Definition

<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" (applicable to all forms of delay variation). However, this metric entry specifies the PDV form defined in section 4.2 of [RFC5481], where the singleton PDV for packet i is referred to by the variable name "PDV(i)".

5.2.2. Fixed Parameters

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

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

Other measurement parameters:

Tmax: a loss threshold waiting time with value 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 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].

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

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

5.3. Method of Measurement

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

5.3.1. Reference Method

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

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

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

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

The reference method requires some way to distinguish between different packets in a stream to establish correspondence between sending times and receiving times for each successfully-arriving packet. Sequence numbers or other send-order identification MUST be

retained at the Src or included with each packet to 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.

5.3.2. Packet Stream Generation

<list of generation parameters and section/spec references 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 Parameter Trunc), and the Src sends each packet at the computed times.

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

Reciprocal_lambda average packet interval for Poisson Streams expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 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]. Reciprocal_lambda = 1 packet per second.

Trunc Upper limit on Poisson distribution expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 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). Trunc = 30.0000 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 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.

5.3.6. Roles

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

Src launches each packet to Dst.

Dst waits for each packet from Src.

5.4. Output

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

5.4.1. 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 one-way delay (undefined delays are excluded), a single value corresponding to the 95th percentile, as follows:

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

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

5.4.2. Reference Definition

<the output type and data format for each type of result>

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

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

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

5.4.3. Metric Units

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

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

5.4.4. Calibration

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

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

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

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

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

5.5. Administrative items

5.5.1. Status

<current or depricated>

5.5.2. Requestor (keep?)

<name of individual or RFC, etc.>

5.5.3. Revision

1.0

5.5.4. Revision Date

YYYY-MM-DD

5.6. Comments and Remarks

<Additional (Informational) details for this entry>

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

6. DNS Response Latency Registry Entry

This section gives an initial registry entry for DNS Response Latency. RFC 2681 [RFC2681] defines a Round-trip delay metric. We build on that metric by specifying several of the input parameters to precisely define a metric for measuring DNS latency.

6.1. Summary

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

<skipping some admin columns for now>

6.1.1. ID (Identifier)

<insert numeric identifier, an integer>

6.1.2. Name

<insert name according to metric naming convention>

RTDNS_Active_IP-UDP-Poisson_RFCXXXXsecY_Seconds_Raw

6.1.3. URI

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

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

6.1.4. Description

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

6.1.5. Change Controller

IETF

6.1.6. Version (of Registry Format)

1.0

6.2. Metric Definition

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

6.2.1. Reference Definition

<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-singleton sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

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

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

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 as defined in Section 13 of [RFC2330]:

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255
  - Protocol: Set to 17 (UDP)
- IPv6 header values:
  - DSCP: set to 0
  - Hop Count: set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Source port: 53
  - Destination port: 53

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

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

  * The DNS header section contains:

    + Identification (see the Run-time column)

    + QR: set to 0 (Query)

    + OPCODE: set to 0 (standard query)

    + AA: not set

    + TC: not set

    + RD: set to one (recursion desired)

    + RA: not set

    + RCODE: not set

    + QDCOUNT: set to one (only one entry)

    + ANCOUNT: not set

    + NSCOUNT: not set

    + ARCOUNT: not set

  * The Question section contains:

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

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

    + QCLASS: set to 1 for IN

  * The other sections do not contain any Resource Records.

Other measurement parameters:

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

* 5.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 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].

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

6.3. Method of Measurement

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

6.3.1. Reference Method

<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 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 reply. Therefore, 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. Sequence number is part of the payload described under Fixed Parameters.

DNS Messages bearing Queries provide for random ID Numbers in the Identification header field, so more than one query may be launched while a previous request is outstanding when the ID Number is used.

IF a DNS response does not arrive within Tmax, the result is undefined. The Message ID SHALL be used to disambiguate the successive queries.

>>> This would require support of ID generation and population in the Message. An alternative would be to use a random Source port on the Query Message, but we would choose ONE before proceding.

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

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

6.3.2. Packet Stream Generation

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

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

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

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

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

6.3.3. Traffic Filtering (observation) Details

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

<section reference>.

NA

6.3.4. Sampling Distribution

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

NA

6.3.5. Run-time Parameters and Data Format

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

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

Src the IP address of 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.)

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

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

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

6.3.6. Roles

<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

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

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

6.4.2. Reference Definition

<describe the data format for each type of result>

For all outputs:

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

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

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

6.4.3. Metric Units

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

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

6.4.4. Calibration

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

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

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

6.5. Administrative items

6.5.1. Status

<current or depricated>

6.5.2. Requestor

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 specifies five initial registry entries for the UDP Poisson One-way Delay.

Note: Each Registry "Name" below specifies a single registry entry, whose output format varies according to the <statistic> element of the name that specifies one form of statistical summary.

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

7.1. Summary

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

7.1.1. ID (Identifier)

<insert numeric identifier, an integer, one corresponding to each name below>

7.1.2. Name

<insert name according to metric naming convention>

OWDelay_Active_IP-UDP-Poisson-Payload250B_RFCXXXXsecY_Seconds_<statistic>

where <statistic> is one of:

- 95Percentile
- Mean
- Min
- Max
- StdDev

7.1.3. URI and URL

URI: Prefix urn:ietf:metrics:perf:<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.

where <statistic> is one of:

- 95Percentile
- Mean

- Min
- Max
- StdDev

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

[RFC7679]

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

[RFC6049]

<specific section reference and additional clarifications, if needed>

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

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

7.2.2. Fixed Parameters

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

Type-P:

- IPv4 header values:

      * DSCP: set to 0

      * TTL: set to 255

      * Protocol: Set to 17 (UDP)

   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 and included in the header

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

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

      * 250 octets total, including the TWAMP format

   Other measurement parameters:

   Tmax: a loss threshold waiting time with value 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 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].

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

7.3. Method of Measurement

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

7.3.1. Reference Method

   <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 [RFC7679] and section 4.6 of [RFC7679] using the Type-P and Tmax defined under Fixed Parameters.

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

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

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

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

7.3.2. Packet Stream Generation

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

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

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

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

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

Reciprocal_lambda average packet interval for Poisson Streams expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 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]. Reciprocal_lambda = 1 packet per second.

Trunc Upper limit on Poisson distribution expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 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). Trunc = 30.0000 seconds.

7.3.3. Traffic Filtering (observation) Details

NA

7.3.4. Sampling Distribution

NA

7.3.5. Run-time Parameters and Data Format

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

<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.

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

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

See subsection titles below for Types.

7.4.2. Reference Definition

<describe the data format for each type of result>

For all output types ---

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

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

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

7.4.2.1. Percentile95

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

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

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

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

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

7.4.2.2. Mean

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

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

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

Mean The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9

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

7.4.2.3. Min

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

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

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

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

7.4.2.4. Max

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

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

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

```
Max = (FiniteDelay [j])

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

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

7.4.2.5. Std_Dev

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

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

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

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

7.4.3. Metric Units

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

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

7.4.4. Calibration

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

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

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

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

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

7.5. Administrative items

7.5.1. Status

<current or depricated>

7.5.2. Requestor (keep?)

name or RFC, etc.

7.5.3. Revision

1.0

7.5.4. Revision Date

YYYY-MM-DD

7.6. Comments and Remarks

Additional (Informational) details for this entry

8. UDP Periodic One-way Delay Registry Entries

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

Note: Each Registry "Name" below specifies a single registry entry, whose output format varies according to the <statistic> element of the name that specifies one form of statistical summary.

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

8.1. Summary

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

8.1.1. ID (Identifier)

<insert numeric identifier, an integer, one corresponding to each name below>

8.1.2. Name

<insert name according to metric naming convention>

OWDelay_Active_IP-UDP-Periodic-Payload142B_RFCXXXXsecY_Seconds_<statistic>

where <statistic> is one of:

- 95Percentile
- Mean
- Min
- Max
- StdDev

8.1.3. URIs

URI: Prefix urn:ietf:metrics:perf:<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.

where <statistic> is one of:

- 95Percentile
- Mean
- Min
- Max
- StdDev

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

[RFC7679]

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

[RFC6049]

<specific section reference and additional clarifications, if needed>

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

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

8.2.2. Fixed Parameters

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

Type-P:

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

Other measurement parameters:

Tmax: a loss threshold waiting time with value 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 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].

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

8.3. Method of Measurement

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

8.3.1. Reference Method

<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 [RFC7679] and section 4.6 of [RFC7679] using the Type-P and Tmax defined under Fixed Parameters.

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

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

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

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

8.3.2. Packet Stream Generation

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

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

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

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

dT the duration of the interval for allowed sample start times

T0 the actual start time of the periodic stream

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

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

8.3.3. Traffic Filtering (observation) Details

NA

8.3.4. Sampling Distribution

NA

8.3.5. Run-time Parameters and Data Format

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

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

Src the IP address of 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.

>>> 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

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

See subsection titles in Data Format for Types.

8.4.2. Reference Definition

<describe the data format for each type of result>

For all output types ---

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

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

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

8.4.2.1. Percentile95

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

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

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

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

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

8.4.2.2. Mean

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

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

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

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

8.4.2.3. Min

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

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

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

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

8.4.2.4. Max

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

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

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

```
Max = (FiniteDelay [j])

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

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

8.4.2.5. Std_Dev

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

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

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

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

8.4.3. Metric Units

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

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

8.4.4. Calibration

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

For one-way delay measurements, the error calibration must include an assessment of the internal clock synchronization with its external reference (this internal clock is supplying timestamps for measurement). In practice, the time offsets of clocks at both the

source and destination are needed to estimate the systematic error due to imperfect clock synchronization (the time offsets are smoothed, thus the random variation is not usually represented in the results).

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

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

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

8.5. Administrative items

8.5.1. Status

<current or depricated>

8.5.2. Requestor (keep?)

name or RFC, etc.

8.5.3. Revision

1.0

8.5.4. Revision Date

YYYY-MM-DD

8.6. Comments and Remarks

Additional (Informational) details for this entry

9. ver08 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.

9.1.1. ID (Identifier)

<insert numeric identifier, an integer>

9.1.2. Name

<insert name according to metric naming convention>

9.1.3. URIs

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

URL:

9.1.4. Description

TBD.

9.1.5. Reference

<reference to the RFC of spec where the registry entry is defined>

9.1.6. Change Controller

<org or person >

9.1.7. Version (of Registry Format)

<currently 1.0>

9.2. Metric Definition

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

9.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

<specific section reference and additional clarifications, if needed>

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>

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 Stream Generation

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

9.3.3. 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>.

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

<list of run-time parameters, and any 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

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

9.4.2. Reference Definition

<pointer to section/spec where output type/format is defined>

9.4.3. Metric Units

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

9.4.4. Calibration

<describe the error calibration, a way to indicate that the results were collected in a calbration mode of operation, and a way to report internal status metrics related to calibration, such as time offset>

9.5. Administrative items

9.5.1. Status

<current or depricated>

9.5.2. Requestor

<name of individual or Internet Draft, 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. 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.

10.1. Registry Indexes

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

10.1.1. Identifier

An integer having enough digits to uniquely identify each entry in the Registry.

10.1.2. Name

A metric naming convention is TBD.

10.1.3. URI

Prefix urn:ietf:metrics:param:<name>

10.1.4. Status

current

10.1.5. Requestor

Alcelip Mornuley

10.1.6. Revision

1.0

10.1.7. Revision Date

2014-07-04

10.1.8. Description

TBD.

10.1.9. Reference Specification(s)

[RFC3611][RFC4566][RFC6776][RFC6792][RFC7003]

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. Section 3.2 of [RFC7003] provides the reference information for this category.

10.2.1. Reference Definition

Packets Discarded in Bursts:

The total number of packets discarded during discard bursts. The measured value is unsigned value. If the measured value exceeds 0xFFFFFD, the value 0xFFFFFE MUST be reported to indicate an over-range measurement. If the measurement is unavailable, the value 0xFFFFFF MUST be reported.

10.2.2. 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.

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. 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].

10.3.1. 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.

10.3.2. 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.

10.3.3. Output Type and Data Format

The output type defines the type of result that the metric produces.

- Value: Packets Discarded in Bursts
- Data Format: 24 bits
- Reference: Section 3.2 of [RFC7003]

10.3.4. Metric Units

The measured results are apparently expressed in packets, although there is no section of [RFC7003] titled "Metric Units".

10.3.5. Run-time Parameters and Data Format

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

10.4. Comments and Remarks

TBD.

11. Revision History

This section may be removed for publication. It contains partial information on updtes.

This draft replaced draft-mornuley-ippm-initial-registry.

In version 02, Section 4 has been edited to reflect recent discussion on the ippm-list: * Removed the combination or "Raw" and left 95th percentile. * Hanging Indent on Run-time parameters (Fixed parameters use bullet lists and other indenting formats. * Payload format for

measurement has been removed. * Explanation of Conditional delay distribution.

Version 03 addressed Phil Eardley’s comments and suggestions in sections 1-4. and resolved the definition of Percentiles.

Version 04 * All section 4 parameters reference YANG types for alternate data formats. * Discussion has concluded that usecase(s) for machine parse-able registry columns are not needed.

12. Security Considerations

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

13. IANA Considerations

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

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

14. 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. 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.

15. References

15.1. 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>.

[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, <http://www.rfc-editor.org/info/rfc7679>.

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

15.2. Informative References

[Brow00] Brownlee, N., "Packet Matching for NeTraMet Distributions", March 2000.

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

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

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

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

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

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

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

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

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

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

[RFC6776] Clark, A. and Q. Wu, "Measurement Identity and Information Reporting Using a Source Description (SDES) Item and an RTCP Extended Report (XR) Block", RFC 6776, DOI 10.17487/RFC6776, October 2012, <http://www.rfc-editor.org/info/rfc6776>.

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

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

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

Authors' Addresses

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

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

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

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

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich
ENGLAND

Email: philip.eardley@bt.com

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

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

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: September 10, 2017

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

March 9, 2017

Registry for Performance Metrics
draft-ietf-ippm-metric-registry-11

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2017.

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

However, despite the importance of Performance Metrics, there are two related problems for the industry. First, how to ensure that when one party requests another party to measure (or report or in some way act on) a particular Performance Metric, then both parties have exactly the same understanding of what Performance Metric is being referred to. Second, how to discover which Performance Metrics have been specified, so as to avoid developing new Performance Metric that is very similar, but not quite inter-operable. The problems can be addressed by creating a registry of performance metrics. The usual way in which IETF organizes namespaces is with Internet Assigned Numbers Authority (IANA) registries, and there is currently no Performance Metrics Registry maintained by the IANA.

This document therefore requests that IANA create and maintain a Performance Metrics Registry, according to the maintenance procedures and the Performance Metrics Registry format defined in this memo. Although the Registry format is primarily for use by IANA, any other organization that wishes to create a Performance Metrics Registry MAY use the same format for their purposes. The authors make no guarantee of the 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 so, 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 [RFC2119].

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

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

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

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 [RFC5226] selected by the IESG to validate the Performance Metrics before updating the Performance Metrics Registry. The Performance Metrics Experts work closely with IANA.

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

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

where we are actually using loss to infer its compliment: delivered throughput.

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

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

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

3. Scope

This document is meant mainly for two different audiences. For those defining new Registered Performance Metrics, it provides specifications and best practices to be used in deciding which Registered Performance Metrics are useful for a measurement study, instructions for writing the text for each column of the Registered Performance Metrics, and information on the supporting documentation required for the new Performance Metrics Registry entry (up to and including the publication of one or more RFCs or I-Ds describing it). For the appointed Performance Metrics Experts and for IANA personnel administering the new IANA Performance Metric 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 organization who are defining a Performance Metric registry of its own, who can rely on the Performance Metric registry defined in this document.

This Performance Metric Registry is applicable to Performance Metrics issued from Active Measurement, Passive Measurement, and any other

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

This document makes no attempt to populate the Performance Metrics Registry with initial entries. It does provides a few examples that are merely illustrations and should not be included in the registry at this point in time.

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

4. Motivation for a Performance Metrics Registry

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

4.1. Interoperability

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

- Control protocol: this type of protocols is used to allow one entity to request another entity to perform a measurement using a specific metric defined by the Performance Metrics Registry. One particular example is the LMAP framework [RFC7594]. Using the LMAP terminology, the Performance Metrics Registry is used in the LMAP Control protocol to allow a Controller to request a measurement task to one or more Measurement Agents. In order to enable this use case, the entries of the Performance Metric Registry must be well enough defined to allow a Measurement Agent implementation to trigger a specific measurement task upon the reception of a control protocol message. This requirement heavily constrains the type of entries that are acceptable for the Performance Metric Registry.

- Report protocol: This type of protocols is used to allow an entity to report measurement results to another entity. By referencing to a specific Performance Metric Registry, it is possible to

properly characterize the measurement result data being reported. Using the LMAP terminology, the Performance Metrics Registry is used in the Report protocol to allow a Measurement Agent to report measurement results to a Collector.

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

4.2. Single point of reference for Performance Metrics

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

4.3. Side benefits

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

Registered Performance Metrics which have undergone such testing SHOULD be noted, with a reference to the test results.

5. Criteria for Performance Metrics Registration

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

1. interpretable by the user.

2. implementable by the software designer,

3. deployable by network operators,

4. accurate, for interoperability and deployment across vendors,

5. Operationally useful, so that it has significant industry interest and/or has seen deployment,

6. Sufficiently tightly defined, so that different values for the Run-time Parameters does not change the fundamental nature of the measurement, nor change the practicality of its implementation.

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

6. Performance Metric Registry: Prior attempt

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

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

1. "It is not believed to be feasible or even useful to register every possible combination of Type P, metric parameters, and Stream parameters using the current structure of the IPPM Metrics Registry."

2. "The registry structure has been found to be insufficiently detailed to uniquely identify IPPM metrics."

3. "Despite apparent efforts to find current or even future users, no one responded to the call for interest in the RFC 4148 registry during the second half of 2010."

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

6.1. Why this Attempt Will Succeed

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

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

7. Definition of the Performance Metric Registry

This Performance Metric Registry is applicable to Performance Metrics used for Active Measurement, Passive Measurement, and any other form of Performance Metric. Each category of measurement has unique properties, so some of the columns defined below are not applicable for a given metric category. In this case, the column(s) SHOULD be

populated with the "NA" value (Non Applicable). However, the "NA" value MUST NOT be used by any metric in the following columns: Identifier, Name, URI, Status, Requester, Revision, Revision Date, Description. In the future, a new category of metrics could require additional columns, and adding new columns is a recognized form of registry extension. The specification defining the new column(s) MUST give guidelines to populate the new column(s) for existing entries (in general).

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

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

Registry Categories and Columns, shown as

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

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

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

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

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

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

7.1. Summary Category

7.1.1. Identifier

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

The Registered Performance Metric unique identifier is a 16-bit integer (range 0 to 65535).

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

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

7.1.2. Name

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

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

MetricType_Method_SubTypeMethod_... Spec_Units_Output

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

  RTDelay (Round Trip Delay)

  RTDNS (Response Time Domain Name Service)

  OWDelay (One Way Delay)

  RTLoss (Round Trip Loss)

  OWLoss (One Way Loss)

  OWPDV (One Way Packet Delay Variation)

  OWIPDV (One Way Inter-Packet Delay Variation)

  OWReorder (One Way Packet Reordering)

  OWDuplic (One Way Packet Duplication)

  OWBTC (One Way Bulk Transport Capacity)

  OWMBM (One Way Model Based Metric)

  SPMonitor (Single Point Monitor)

  MPMonitor (Multi-Point Monitor)

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

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

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

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

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

Spatial (Spatial Metric of RFC5644)

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

  ICMP (Internet Control Message Protocol)

  IP (Internet Protocol)

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

  UDP (User Datagram Protocol)

  TCP (Transport Control Protocol)

  Poisson (Packet generation using Poisson distribution)

  Periodic (Periodic packet generation)

  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)

  @@@@<add others from MBM draft?>

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

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

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

  Seconds

  RatioPercent (value multiplied by 100)

  BPS (Bits per Second)

  EventTotal (for unit-less counts)

  Multiple (more than one type of unit)

  Enumerated (a list of outcomes)

  Unitless

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

  Singleton (sometimes called raw data)

  Minimum

  Maximum

  Median

  Mean

  95Percentile (95th Percentile)

  99Percentile (99th Percentile)

  StdDev (Standard Deviation)

  Variance

  PFI (Pass, Fail, Inconclusive)

  FlowRecords (descriptions of flows observed)

An example is:

RTDelay_Active_IP-UDP-Poisson_RFCXXXXsecY_Seconds_95percentile

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

Note that private registries following the format described here SHOULD use the prefix "Priv_" on any name to avoid unintended

conflicts (further considerations are described in section 10). Private registry entries usually have no specifying RFC, thus the Spec: element has no clear interpretation.

7.1.3. URIs

The URIs column MUST contain a URI [RFC3986] that uniquely identifies the metric. This URI is a URN [RFC2141]. The URI is automatically generated by prepending the prefix

urn:ietf:metrics:perf:

to the metric name. The resulting URI is globally unique.

The URIs column MUST contain a second URI which is a URL [RFC3986] and uniquely identifies and locates the metric entry so it is accessible through the Internet. The URL points to a file containing the human-readable information of exactly one registry entry. Ideally, the file will be HTML-formated and contain URLs to referenced sections of HTML-ized RFCs. The separate files for different entries can be more easily edited and re-used when preparing new entries. The exact composition of each metric URL will be determined by IANA and reside on "iana.org", but there will be some overlap with the URN described above. The major sections of [I-D.ietf-ippm-initial-registry] provide an example 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 revsions to the regsitry entry, and provides contact information.

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.

7.2. Metric Definition Category

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

7.2.1. Reference Definition

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

7.2.2. Fixed Parameters

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

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

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

Parameters MUST have a well-specified data format.

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

7.3. Method of Measurement Category

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

7.3.1. Reference Method

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

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

7.3.2. Packet Stream Generation

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

Each entry for this column contains the following information:

- Value: The name of the packet stream scheduling discipline
- Reference: the specification where the stream is defined

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

The simplest example of stream specification is Singleton scheduling (see [RFC2330]), where a single atomic measurement is conducted. Each atomic measurement could consist of sending a single packet (such as a DNS request) or sending several packets (for example, to request a webpage). Other streams support a series of atomic measurements in a "sample", with a schedule defining the timing between each transmitted packet and subsequent measurement. Principally, two different streams are used in IPPM metrics, Poisson

distributed as described in [RFC2330] and Periodic as described in [RFC3432]. Both Poisson and Periodic have their own unique parameters, and the relevant set of parameters names and values should be included either in the Fixed Parameters column or in the Run-time parameter column.

7.3.3. Traffic Filter

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

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

The traffic filter includes the following information:

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

Value: the actual set of rules expressed

7.3.4. Sampling Distribution

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

Value: the name of the sampling distribution

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

The sampling distribution may require parameters. In case such parameters are needed, they should be included either in the Fixed

parameter column or in the run time parameter column, depending on wether they will be fixed or will be an input for the metric.

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

7.3.5. Run-time Parameters

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

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

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

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

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

7.3.6. Role

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

this particular entry. In the previous example, there should be two entries in the registry, one for each role, so that when a measurement agent is instructed to perform the one way delay source metric know that it is supposed to generate packets. The values for this field are defined in the reference method of measurement.

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

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

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

7.5. Administrative information

7.5.1. Status

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

7.5.2. Requester

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

7.5.3. Revision

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

7.5.4. Revision Date

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

7.6. Comments and Remarks

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

8. The Life-Cycle of Registered Performance Metrics

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

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

8.1. Adding new Performance Metrics to the Performance Metrics Registry

Requests to change Registered Performance Metrics in the Performance Metric Registry are submitted to IANA, which forwards the request to a designated group of experts (Performance Metric Experts) appointed by the IESG; these are the reviewers called for by the Expert Review RFC5226 policy defined for the Performance Metric Registry. The Performance Metric Experts review the request for such things as compliance with this document, compliance with other applicable Performance Metric-related RFCs, and consistency with the currently defined set of Registered Performance Metrics.

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

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

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

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

8.2. Revising Registered Performance Metrics

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

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

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

Revisions are initiated by sending a candidate Registered Performance Metric definition to IANA, as in Section 8, identifying the existing Performance Metrics Registry entry.

The primary requirement in the definition of a policy for managing changes to existing Registered Performance Metrics is avoidance of interoperability problems; Performance Metric Experts must work to maintain interoperability above all else. Changes to Registered Performance Metrics may only be done in an inter-operable way; necessary changes that cannot be done in a way to allow interoperability with unchanged implementations must result in the creation of a new Registered Performance Metric and possibly the deprecation of the earlier metric.

A change to a Registered Performance Metric is held to be backward-compatible only when:

1. "it involves the correction of an error that is obviously only editorial; or"

2. "it corrects an ambiguity in the Registered Performance Metric's definition, which itself leads to issues severe enough to prevent the Registered Performance Metric's usage as originally defined; or"

3. "it corrects missing information in the metric definition without changing its meaning (e.g., the explicit definition of 'quantity' semantics for numeric fields without a Data Type Semantics value); or"

4. "it harmonizes with an external reference that was itself corrected."

If an Performance Metric revision is deemed permissible by the Performance Metric Experts, according to the rules in this document, IANA makes the change in the Performance Metric Registry. The requester of the change is appended to the requester in the Performance Metrics Registry.

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

Where applicable, additions to 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 is 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 Revising Registered Performance Metrics; or"

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

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

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

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

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

10. IANA Considerations

This document requests the following IANA Actions.

10.1. New Namespace Assignments

This document requests the allocation of the URI prefix urn:ietf:metrics for the purpose of generating URIs for metrics in general. The registration procedure for the new "metrics" URN sub-namespace is IETF Review.

This document requests the allocation of the URI prefix urn:ietf:metrics:perf for the purpose of generating URIs for Registered Performance Metrics. The registration procedures for the new "perf" URN sub-namespace are Expert Review or IETF Standards

Action, and coordinated with the entries added to the New Performance Metrics Registry (see below).

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 "IETF URN Sub-namespace for Registered Performance Metric Name Elements" (urn:ietf:metrics:perf). Each Registry, whose names are listed below:

MetricType:

Method:

SubTypeMethod:

Spec:

Units:

Output:

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

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

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

A candidate Metric Entry RFC or document for Expert Review would propose one or more new element values required to describe the unique entry, and the new name element(s) would be reviewed along with the metric entry. New assignments for IETF URN Sub-namespace for Registered Performance Metric Name Elements will be administered by IANA through Expert Review [RFC5226], 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 "Registered Performance Metrics". This Registry will contain the following Summary columns:

Identifier:

Name:

URIs:

Description:

Reference:

Change Controller:

Version:

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

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

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

The "URIs" column will have a URL to the full template of each registry entry, and the linked text may be the URN itself. The template shall be HTML-ized to aid the reader, with links to reference RFCs (similar to the way that Internet Drafts are HTML-ized, the same tool can perform the function).

The "Reference" column will include an RFC, an approved specification from another standards body, or the contact person.

New assignments for Performance Metric Registry will be administered by IANA through Expert Review [RFC5226], i.e., review by one of a group of experts, the Performance Metric Experts, who are appointed by the IESG upon recommendation of the Transport Area Directors. 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 Metric Registry require IETF Standards Action. Only one form of registry extension is envisaged:

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

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

11. Acknowledgments

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

12. References

12.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <http://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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, DOI 10.17487/RFC2141, May 1997, <http://www.rfc-editor.org/info/rfc2141>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <http://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, <http://www.rfc-editor.org/info/rfc3986>.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[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>.

[RFC6576] Geib, R., Ed., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, DOI 10.17487/RFC6576, March 2012, <http://www.rfc-editor.org/info/rfc6576>.

12.2. Informative 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>.

[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, <http://www.rfc-editor.org/info/rfc7679>.

[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>.

[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>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc3611>.

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

[RFC5474] Duffield, N., Ed., Chiou, D., Claise, B., Greenberg, A., Grossglauser, M., and J. Rexford, "A Framework for Packet Selection and Reporting", RFC 5474, DOI 10.17487/RFC5474, March 2009, <http://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, <http://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, <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>.

[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>.

[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, <http://www.rfc-editor.org/info/rfc6035>.

[RFC6776] Clark, A. and Q. Wu, "Measurement Identity and Information Reporting Using a Source Description (SDES) Item and an RTCP Extended Report (XR) Block", RFC 6776, DOI 10.17487/RFC6776, October 2012, <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>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7594>.

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

[I-D.ietf-ippm-initial-registry] Morton, A., Bagnulo, M., Eardley, P., and K. D'Souza, "Initial Performance Metric Registry Entries", draft-ietf-ippm-initial-registry-02 (work in progress), October 2016.

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

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

IPPM WG
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2017

R. Civil
Ciena Corporation
A. Morton
AT&T Labs
R. Rahman
M. Jethanandani
Cisco Systems
K. Pentikousis, Ed.
Travelping
February 22, 2017

Two-Way Active Measurement Protocol (TWAMP) Data Model
draft-ietf-ippm-twamp-yang-03

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 August 26, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Two-Way Active Measurement Protocol (TWAMP) [RFC5357] is used to measure network performance parameters such as latency, bandwidth, and packet loss by sending probe packets and measuring their experience in the network. To date, TWAMP implementations do not come with a standard management framework and, as such, configuration depends on 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 standardizing 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 left unspecified and may be proprietary protocols.

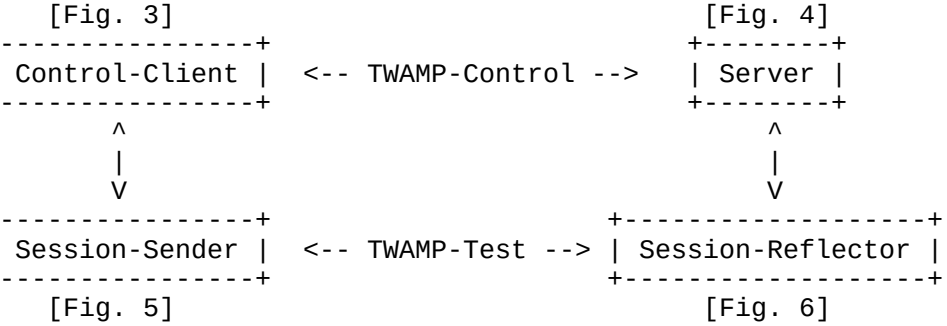


Figure 1: Annotated TWAMP logical model

As per [RFC5357], a TWAMP implementation may follow a simplified logical model, in which the same node acts both as Control-Client and Session-Sender, while another node acts at the same time as TWAMP Server and Session-Reflector. Figure 2 illustrates this simplified logical model and indicates the interaction between the TWAMP configuration client and server using, for instance, NETCONF [RFC6241] or RESTCONF [I-D.ietf-netconf-restconf].

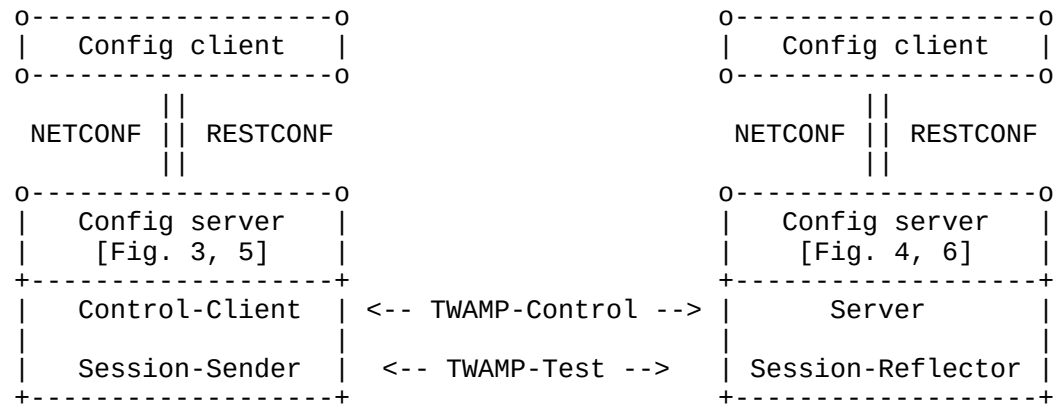


Figure 2: Simplified TWAMP model and protocols

The data model defined in this document is orthogonal to the specific protocol used between the Config client and Config server to communicate the TWAMP configuration parameters.

Operational actions such as how TWAMP-Test sessions are started and stopped, how performance measurement results are retrieved, or how stored results are cleared, and so on, are not addressed by the configuration model defined in this document. As noted above, such operational actions are not part of the TWAMP specification [RFC5357] and hence are out of scope of this document. See also Appendix B.

3. Data Model Overview

The TWAMP data model includes four categories of configuration items.

First, global configuration items relate to parameters that are set on a per device level. For example, the administrative status of the device with respect to whether it allows TWAMP sessions and, if so, in what capacity (e.g. Control-Client, Server or both), is a typical instance of a global configuration item.

A second category includes attributes that can be configured on a per TWAMP-Control connection basis, such as the Server IP address.

A third category includes attributes related to per TWAMP-Test session attributes, for instance setting different values in the Differentiated Services Code Point (DSCP) field.

Finally, the data model includes attributes that relate to the operational state of the TWAMP implementation.

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

3.1. Control-Client

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

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

- A name which can be used to uniquely identify at the Control-Client a particular control connection. This name is necessary for programmability reasons because at the time of creation of a TWAMP-Control connection not all IP and TCP port number information needed to uniquely identify the connection is available.
- The IP address of the interface the Control-Client will use for connections.
- The IP address of the remote TWAMP Server.
- Authentication and encryption attributes such as KeyID, Token and the Client Initialization Vector (Client-IV); see also the last paragraph of Section 6 in [RFC4656] and [RFC4086].

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

- The test session name that uniquely identifies a particular test session at the Control-Client and Session-Sender. Similarly to the control connections above, this unique test session name is needed because at the time of creation of a TWAMP-Test session, for example, the source UDP port number is not known to uniquely identify the test session.
- The IP address and UDP port number of the Session-Sender on the path under test by TWAMP.
- The IP address and UDP port number of the Session-Reflector on said path.

- Information pertaining to the test packet stream, such as the test starting time, which performance metric is to be used [I-D.ietf-ippm-metric-registry], or whether the test should be repeated.

3.2. Server

Each TWAMP Server has an administrative status field set at the device level to indicate whether the node is enabled to function as a TWAMP Server.

Each Server is associated with zero or more TWAMP-Control connections. Each control connection is uniquely identified by the 4-tuple {Control-Client IP address, Control-Client TCP port number, Server IP address, Server TCP port}. Control connection configuration items on a TWAMP Server are read-only.

3.3. Session-Sender

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

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

- The test session name that MUST be identical with the corresponding test session name on the TWAMP Control-Client (Section 3.1).

- The control connection name, which along with the test session name uniquely identify the TWAMP Session-Sender instance.

- Information pertaining to the test packet stream, such as, for example, the number of test packets and the packet distribution to be employed; see also [RFC3432].

3.4. Session-Reflector

Each TWAMP Session-Reflector has an administrative status field set at the device level to indicate whether the node is enabled to function as such.

Each Session-Reflector is associated with zero or more TWAMP-Test sessions. For each test session, the REFWAIT timeout parameter which determines whether to discontinue the session if no packets have been received ([RFC5357], Section 4.2) can be configured.

Read-only access to other data model parameters, such as the Sender IP address is foreseen. Each test session can be uniquely identified by the 4-tuple mentioned in Section 3.2.

4. Data Model Parameters

This section defines the TWAMP data model using UML and introduces selected parameters associated with the four TWAMP logical entities. The complete TWAMP data model specification is provided in the YANG module presented in Section 5.2.

4.1. Control-Client

The client container (see Figure 3) holds items that are related to the configuration of the TWAMP Control-Client logical entity (recall Figure 1).

The client container includes an administrative configuration parameter (client/admin-state) that indicates whether the device is allowed to initiate TWAMP-Control connections.

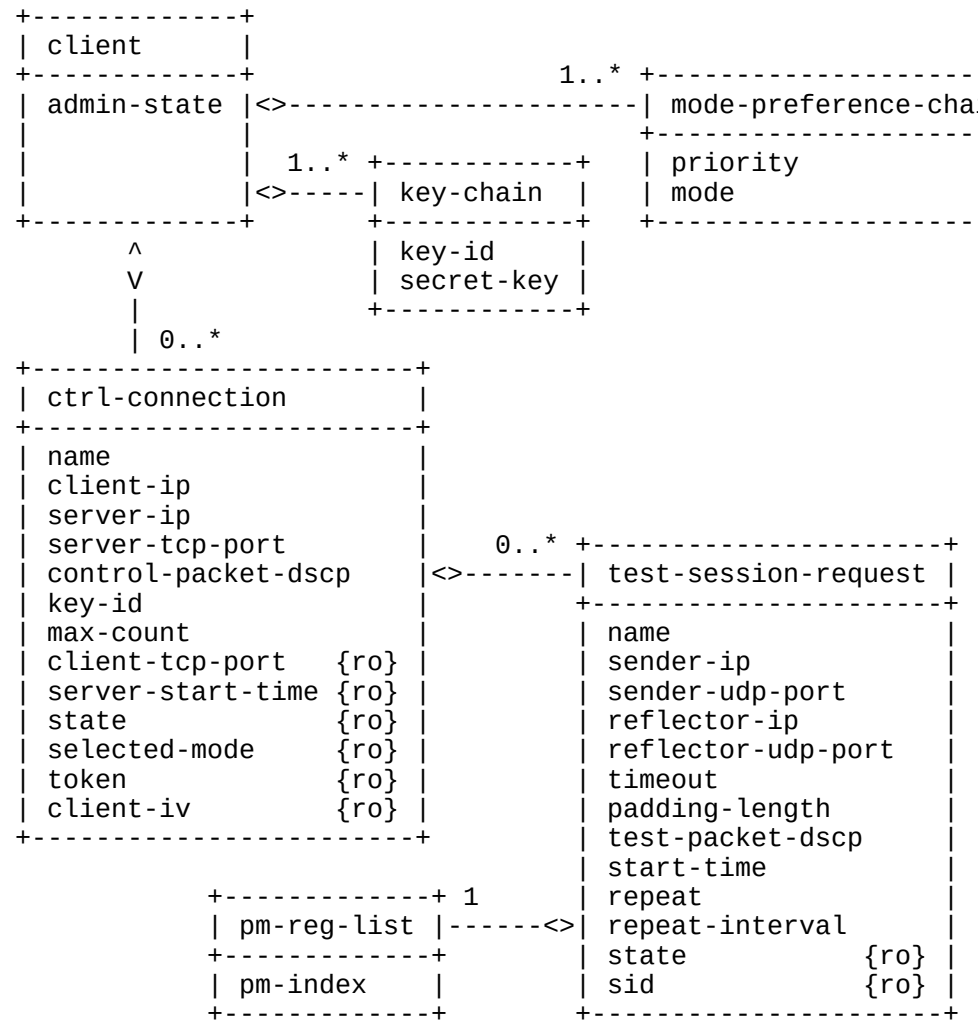


Figure 3: TWAMP Control-Client UML class diagram

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

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], [RFC6038], and [RFC7717]. 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 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 16 octets for the AES Session-key used for encryption and 32 octets for the HMAC-SHA1 Session-key used for authentication; see also Section 6.10 of [RFC4656].

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

In turn, each ctrl-connection holds a list of test-session-request. test-session-request holds information associated with the Control-Client for this test session. This includes information associated with the Request-TW-Session/Accept-Session message exchange (see Section 3.5 of [RFC5357]).

There SHALL be one instance of test-session-request for each TWAMP-Test session that is to be negotiated by this TWAMP-Control connection via a Request-TW-Session/Accept-Session exchange.

The Control-Client is also responsible for scheduling TWAMP-Test sessions so test-session-request holds information related to these actions (e.g. pm-index, repeat-interval).

4.2. Server

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

The server container includes an administrative configuration parameter (server/admin-state) that indicates whether the device is allowed to receive TWAMP-Control connections.

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

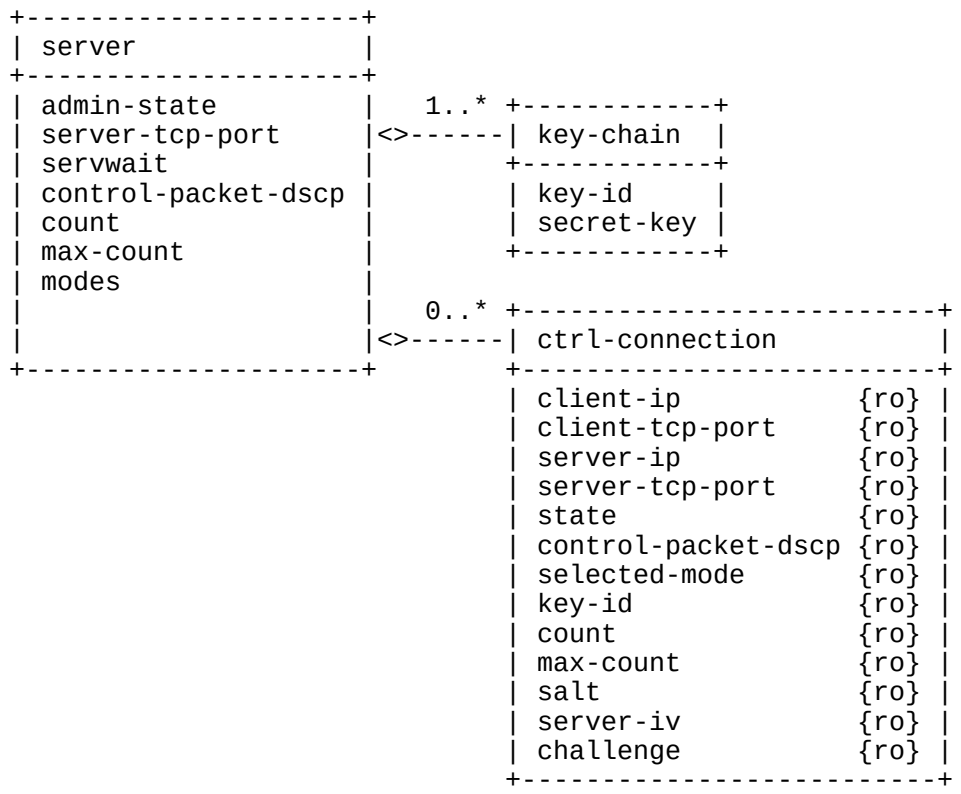


Figure 4: TWAMP Server UML class diagram

Each server container holds a list named key-chain which relates 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 active on the Server is represented by a ctrl-connection. There SHALL be one ctrl-connection per incoming TWAMP-Control (TCP) connection that is received and active on the Server. Each ctrl-connection can be uniquely identified by the 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port}. All items in the ctrl-connection list are read-only.

4.3. Session-Sender

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

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

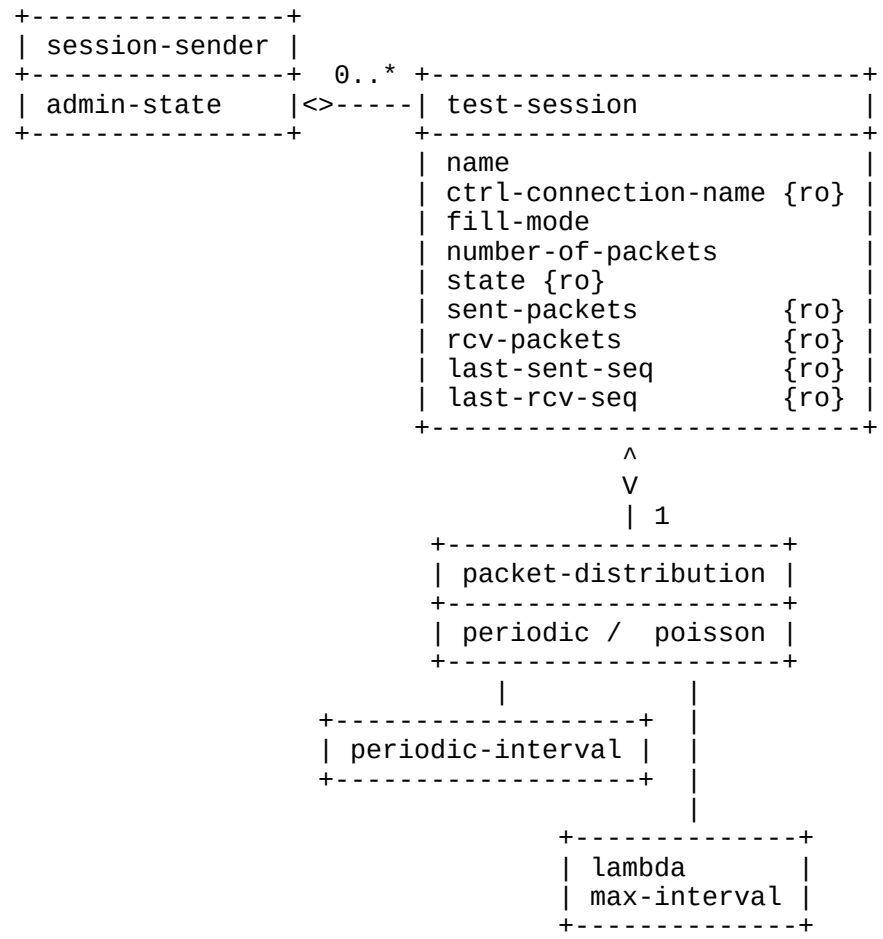


Figure 5: TWAMP Session-Sender UML class diagram

Each TWAMP-Test session initiated by the Session-Sender will be represented by an instance of a test-session object. There SHALL be one instance of test-session for each TWAMP-Test session for which packets are being sent.

4.4. Session-Reflector

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

The session-reflector container includes an administrative parameter (session-reflector/admin-state) that controls whether the device is allowed to respond to incoming TWAMP-Test sessions.

A device operating in the Session-Reflector role cannot configure attributes on a per-session basis, as it has no foreknowledge of what incoming sessions it will receive. As such, any parameter that the Session-Reflector might want to apply to an incoming TWAMP-Test session must be configured at the overall Session-Reflector level and are applied to all incoming sessions.

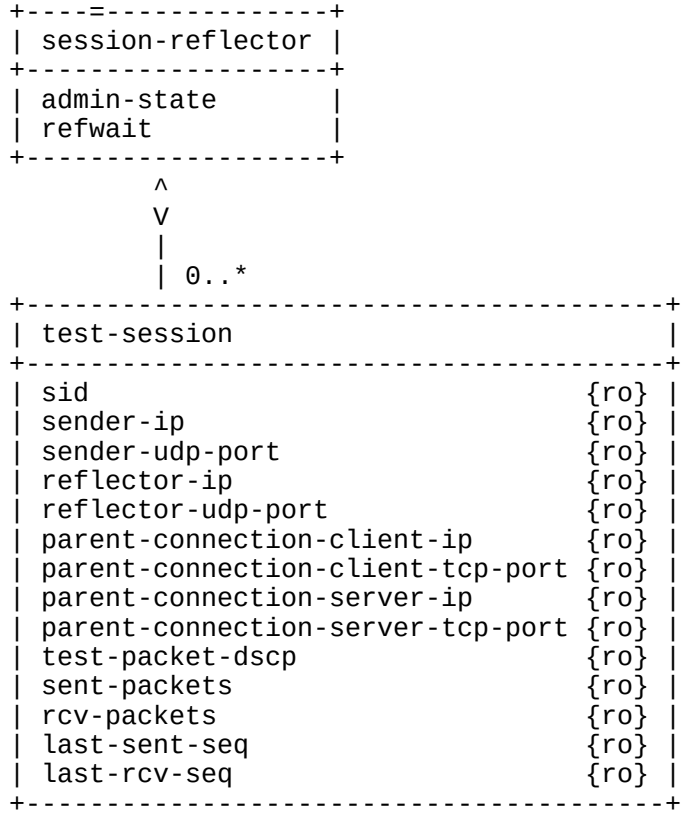


Figure 6: TWAMP Session-Reflector UML class diagram

Each incoming TWAMP-Test session that is active on the Session-Reflector SHALL be represented by an instance of a test-session object. All items in the test-session object are read-only.

Instances of test-session are indexed by a session identifier (sid). This value is auto-allocated by the TWAMP Server as test session

requests are received, and communicated back to the Control-Client in the SID field of the Accept-Session message; see Section 4.3 of [RFC6038].

When attempting to retrieve operational data for active test sessions from a Session-Reflector device, the user will not know what sessions are currently active on that device, or what SIDs have been auto-allocated for these test sessions. If the user has network access to the Control-Client device, then it is possible to read the data for this session under client/ctrl-connection/test-session-request/sid and obtain the SID (see Figure 3). The user may then use this SID value as an index to retrieve an individual session-reflector/test-session instance on the Session-Reflector device.

If the user has no network access to the Control-Client device, then the only option is to retrieve all test-session instances from the Session-Reflector device. This could be problematic if a large number of test sessions are currently active on that device.

Each Session-Reflector TWAMP-Test session contains the following 4-tuple: {parent-connection-client-ip, parent-connection-client-tcp-port, parent-connection-server-ip, parent-connection-server-tcp-port}. This 4-tuple MUST correspond to the equivalent 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port} in server/ctrl-connection. This 4-tuple allows the user to trace back from the TWAMP-Test session to the (parent) TWAMP-Control connection that negotiated this test session.

5. Data Model

This section formally specifies the TWAMP data model using YANG.

5.1. YANG Tree Diagram

This section presents a simplified graphical representation of the TWAMP data model using a YANG tree diagram. Readers should keep in mind that the limit of 72 characters per line forces us to introduce artificial line breaks in some tree diagram nodes.

```
 module: ietf-twamp
     +--rw twamp
        +--rw client! {control-client}?
        |  +--rw admin-state              boolean
        |  +--rw mode-preference-chain* [priority]
        |  |  +--rw priority    uint16
        |  |  +--rw mode?       twamp-modes
        |  +--rw key-chain* [key-id]
        |  |  +--rw key-id        binary
```

```
|  |  +--rw secret-key?   binary
|  +--rw ctrl-connection* [name]
|     +--rw name                    string
|     +--rw client-ip?              inet:ip-address
|     +--rw server-ip               inet:ip-address
|     +--rw server-tcp-port?        inet:port-number
|     +--rw control-packet-dscp?    inet:dscp
|     +--rw key-id?                 string
|     +--rw max-count?              uint8
|     +--ro client-tcp-port?        inet:port-number
|     +--ro server-start-time?      uint64
|     +--ro state? \
|        control-client-connection-state
|     +--ro selected-mode?          twamp-modes
|     +--ro token?                  binary
|     +--ro client-iv?              binary
|     +--rw test-session-request* [name]
|        +--rw name                  string
|        +--rw sender-ip?            inet:ip-address
|        +--rw sender-udp-port?      union
|        +--rw reflector-ip          inet:ip-address
|        +--rw reflector-udp-port?   dynamic-port-number
|        +--rw timeout?              uint64
|        +--rw padding-length?       uint32
|        +--rw test-packet-dscp?     inet:dscp
|        +--rw start-time?           uint64
|        +--rw repeat?               union
|        +--rw repeat-interval?      uint32
|        +--rw pm-reg-list* [pm-index]
|        |  +--rw pm-index    uint16
|        +--ro state?                test-session-state
|        +--ro sid?                  string
+--rw server! {server}?
|  +--rw admin-state            boolean
|  +--rw server-tcp-port?       inet:port-number
|  +--rw servwait?              uint32
|  +--rw control-packet-dscp?   inet:dscp
|  +--rw count?                 uint8
|  +--rw max-count?             uint8
|  +--rw modes?                 twamp-modes
|  +--rw key-chain* [key-id]
|  |  +--rw key-id        binary
|  |  +--rw secret-key?   binary
|  +--ro ctrl-connection* \
|     [client-ip client-tcp-port server-ip server-tcp-port]
|     +--ro client-ip              inet:ip-address
|     +--ro client-tcp-port        inet:port-number
|     +--ro server-ip              inet:ip-address
```

```
|     +--ro server-tcp-port        inet:port-number
|     +--ro state?                 server-ctrl-connection-state
|     +--ro control-packet-dscp?   inet:dscp
|     +--ro selected-mode?         twamp-modes
|     +--ro key-id?                string
|     +--ro count?                 uint8
|     +--ro max-count?             uint8
|     +--ro salt?                  binary
|     +--ro server-iv?             binary
|     +--ro challenge?             binary
+--rw session-sender! {session-sender}?
|  +--rw admin-state     boolean
|  +--rw test-session* [name]
|     +--rw name                      string
|     +--ro ctrl-connection-name?     string
|     +--rw fill-mode?                padding-fill-mode
|     +--rw number-of-packets         uint32
|     +--rw (packet-distribution)?
|     |  +--:(periodic)
|     |  |  +--rw periodic-interval       decimal64
|     |  +--:(poisson)
|     |     +--rw lambda                 decimal64
|     |     +--rw max-interval?          decimal64
|     +--ro state?                    sender-session-state
|     +--ro sent-packets?             uint32
|     +--ro rcv-packets?              uint32
|     +--ro last-sent-seq?            uint32
|     +--ro last-rcv-seq?             uint32
+--rw session-reflector! {session-reflector}?
   +--rw admin-state     boolean
   +--rw refwait?        uint32
   +--ro test-session* \
      [sender-ip sender-udp-port \
       reflector-ip reflector-udp-port]
      +--ro sid?                                  string
      +--ro sender-ip                             inet:ip-address
      +--ro sender-udp-port \
         dynamic-port-number
      +--ro reflector-ip                          inet:ip-address
      +--ro reflector-udp-port \
         dynamic-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 test-packet-dscp?                     inet:dscp
```

```
              +--ro sent-packets?                        uint32
              +--ro rcv-packets?                         uint32
              +--ro last-sent-seq?                       uint32
              +--ro last-rcv-seq?                        uint32
```

Figure 7: YANG Tree Diagram.

5.2. YANG Module

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

```
<CODE BEGINS> file "ietf-twamp@2017-02-22.yang"
module ietf-twamp {
  namespace
    urn:ietf:params:xml:ns:yang:ietf-twamp;

  prefix
    ietf-twamp;

  import ietf-inet-types {
    prefix inet;
  }

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

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

  description
    "This YANG module specifies a vendor-independent data
    model for the Two-Way Active Measurement Protocol (TWAMP).

    The data model covers four TWAMP logical entities, namely,
    Control-Client, Server, Session-Sender, and Session-Reflector,
    as illustrated in the annotated TWAMP logical model (Fig. 1
    of draft-ietf-ippm-twamp-yang).

    This YANG module uses features to indicate which of the four
    logical entities are supported by a TWAMP implementation.";

  revision 2017-02-22 {
    description
      "Revision appearing in draft-ietf-ippm-twamp-yang-03.

      Covers RFC 5357, RFC 5618, RFC 5938, RFC 6038, RFC 7717, and
      draft-ietf-ippm-metric-registry";
```

```
    reference
      draft-ietf-ippm-twamp-yang;
  }


  /*
   * Typedefs
   */

  typedef twamp-modes {
    type bits {
      bit unauthenticated {
        position 0;
        description
          "Unauthenticated mode, in which no encryption or
          authentication is applied in TWAMP-Control and TWAMP-Test.
          KeyID, Token, and Client-IV are not used in the
          Set-Up-Response message. See Section 3.1 of RFC 4656.";
        reference
          "RFC 4656: A One-way Active Measurement Protocol (OWAMP)";
      }
      bit authenticated {
        position 1;
        description
          "Authenticated mode, in which the Control-Client and Server
          possess a shared secret thus prohibiting 'theft of service'.
          As per Section 6 of RFC 4656, in 'authenticated mode, the
          timestamp is in the clear and is not protected
          cryptographically in any way, while the rest of the message
          has the same protection as in encrypted mode. This mode
          allows one to trade off cryptographic protection against
          accuracy of timestamps.'";
        reference
          "RFC 4656: A One-way Active Measurement Protocol (OWAMP)";
      }
      bit encrypted {
        position 2;
        description
          "Encrypted mode 'makes it impossible to alter
          timestamps undetectably.' See also Section 4 of RFC 7717
          and Section 6 of RFC 4656.";
        reference
          "RFC 4656: A One-way Active Measurement Protocol (OWAMP)";
      }
      bit unauth-test-encrpyt-control {
        position 3;
        description
          "When using the Mixed Security Mode, the TWAMP-Test
```

```
          protocol follows the Unauthenticated mode and the
          TWAMP-Control protocol the Encrypted mode.";
        reference
          "RFC 5618: Mixed Security Mode for the Two-Way Active
          Measurement Protocol (TWAMP)";
      }
      bit individual-session-control {
        position 4;
        description
          "This mode enables individual test sessions using
          Session Identifiers.";
        reference
          "RFC 5938: Individual Session Control Feature
          for the Two-Way Active Measurement Protocol (TWAMP)";
      }
      bit reflect-octets {
        position 5;
        description
          "This mode indicates the reflect octets capability.";
        reference
          "RFC 6038: Two-Way Active Measurement Protocol (TWAMP)
          Reflect Octets and Symmetrical Size Features";
      }
      bit symmetrical-size {
        position 6;
        description
          "This mode indicates support for the symmetrical size
          sender test packet format.";
        reference
          "RFC 6038: Two-Way Active Measurement Protocol (TWAMP)
          Reflect Octets and Symmetrical Size Features";
      }
      bit IKEv2Derived {
        position 7;
        description
          "In this mode the the shared key is derived
          from an IKEv2 security association (SA).";
        reference
          "RFC 7717: IKEv2-Derived Shared Secret Key for
          the One-Way Active Measurement Protocol (OWAMP)
          and Two-Way Active Measurement Protocol (TWAMP)";
      }
    }
    description
      "Specifies the configurable TWAMP-Modes supported during a
      TWAMP-Control Connection setup between a Control-Client
      and a Server. Section 7 of RFC 7717 summarizes the
      TWAMP-Modes registry and points to their formal
```

```
      specification.";
  }

  typedef control-client-connection-state {
    type enumeration {
      enum active {
        description
          "Indicates an active TWAMP-Control connection to Server.";
      }
      enum idle {
        description
          "Indicates an idle TWAMP-Control connection to Server.";
      }
    }
    description
      "Indicates the Control-Client TWAMP-Control connection state.";
  }

  typedef test-session-state {
    type enumeration {
      enum accepted {
        value 0;
        description
          "Indicates that accepted TWAMP-Test session request.";
      }
      enum failed {
        value 1;
        description
          "Indicates a TWAMP-Test session failure due to
          some unspecified reason (catch-all).";
      }
      enum internal-error {
        value 2;
        description
          "Indicates a TWAMP-Test session failure due to
          an internal error.";
      }
      enum not-supported {
        value 3;
        description
          "Indicates a TWAMP-Test session failure because
          some aspect of the TWAMP-Test session request
          is not supported.";
      }
      enum permanent-resource-limit {
        value 4;
        description
          "Indicates a TWAMP-Test session failure due to
```

```
          permanent resource limitations.";
      }
      enum temp-resource-limit {
        value 5;
        description
          "Indicates a TWAMP-Test session failure due to
          temporary resource limitations.";
      }
    }
    description
      "Indicates the Control-Client TWAMP-Test session state.";
  }

  typedef server-ctrl-connection-state {
    type enumeration {
      enum active {
        description
          "Indicates an active TWAMP-Control connection
          to the Control-Client.";
      }
      enum servwait {
        description
          "Indicates that the TWAMP-Control connection to the
          Control-Client is in SERVWAIT as per the definition of
          Section 3.1 of RFC 5357.";
      }
    }
    description
      "Indicates the Server TWAMP-Control connection state.";
  }

  typedef sender-session-state {
    type enumeration {
      enum active {
        description
          "Indicates that the TWAMP-Test session is active.";
      }
      enum failure {
        description
          "Indicates that the TWAMP-Test session has failed.";
      }
    }
    description
      "Indicates the Session-Sender TWAMP-Test session state.";
  }

  typedef padding-fill-mode {
    type enumeration {
```

```
      enum zero {
        description
          "TWAMP-Test packets are padded with all zeros.";
      }
      enum random {
        description
          "TWAMP-Test packets are padded with pseudo-random
          numbers.";
      }
    }
    description
      "Indicates what type of packet padding is used in the
      TWAMP-Test packets.";
  }

  typedef dynamic-port-number {
    type inet:port-number {
      range 49152..65535;
    }
    description "Dynamic range for port numbers.";
  }

  /*
   * Features
   */

  feature control-client {
    description
      "Indicates that the device supports configuration of the
      TWAMP Control-Client logical entity.";
  }

  feature server {
    description
      "Indicates that the device supports configuration of the
      TWAMP Server logical entity.";
  }

  feature session-sender {
    description
      "Indicates that the device supports configuration of the
      TWAMP Session-Sender logical entity.";
  }

  feature session-reflector {
    description
      "Indicates that the device supports configuration of the
      TWAMP Session-Reflector logical entity.";
```

```
  }


  /*
   * Reusable node groups
   */

  grouping key-management {
    list key-chain {
      key key-id;
      leaf key-id {
        type binary {
          length 1..80;
        }
        description
          "KeyID used for a TWAMP-Control connection. As per
          Section 3.1 of RFC 4656, KeyID is 'a UTF-8 string, up to
          80 octets in length' and is used to select which 'shared
          shared secret the [Control-Client] wishes to use to
          authenticate or encrypt'.";
        }
        leaf secret-key {
          type binary;
          description
            "The secret key corresponding to the KeyID for this
            TWAMP-Control connection.";
        }
        description
          "Relates KeyIDs with their respective secret keys
          in a TWAMP-Control connection.";
    }
    description
        "Used by the Control-Client and Server for TWAMP-Control
        key management.";
  }

  grouping maintenance-statistics {
    leaf sent-packets {
      type uint32;
      config false;
      description "Indicates the number of packets sent.";
    }

    leaf rcv-packets {
      type uint32;
      config false;
      description "Indicates the number of packets received.";
    }
```

```
    leaf last-sent-seq {
      type uint32;
      config false;
      description "Indicates the last sent sequence number.";
    }

    leaf last-rcv-seq {
      type uint32;
      config false;
      description "Indicates the last received sequence number.";
    }
    description "Used for TWAMP-Test maintenance statistics.";
  }

  grouping count {
    leaf count {
      type uint8 {
        range "10..31";
      }
      default 10;
      description
        "Parameter communicated to the Control-Client as part of the
        Server Greeting message and used for deriving a key from a
        shared secret as per Section 3.1 of  RFC 4656:  MUST be a
        power of 2 and at least 1024; SHOULD be increased as more
        computing power becomes common.";
      }
   description
     "Reusable data structure for count which is used both in the
     Server container.";
  }

  grouping max-count {
    leaf max-count {
      type uint8 {
        range "10..31";
      }
      default 15;
      description
        "This parameter limits the maximum Count value, which MUST
        be a power of 2 and at least 1024 as per RFC 5357. It is
        configured by providing said power. For example,
        configuring 10 here means max count 2^10 = 1024.
        The default is 15, meaning 2^15 = 32768.

        A TWAMP Server uses this configured value in the
        Server-Greeting message sent to the Control-Client.
```

```
        A TWAMP Control-Client uses this configured value to prevent
        denial-of-service (DOS) attacks by closing the control
        connection to the Server if it 'receives a Server-Greeting
        message with Count greater that its maximum configured value',
        as per Section 6 of RFC 5357.

        Further, note that according to Section 6 of RFC 5357:

          'If an attacking system sets the maximum value in
          Count (2**32), then the system under attack would stall
          for a significant period of time while it attempts to
          generate keys.

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

        RFC 5357 does not qualify 'significant period' in terms of
        time, but it is clear that this depends on the processing
        capacity available and operators need to pay attention to
        this security consideration.";
      }
    description
      "Reusable data structure for max-count which is used both at
      the Control-Client and the Server containers.";
  }


  /*
   * Configuration data nodes
   */

  container twamp {
    description
        "TWAMP logical entity configuration grouping of four models
        which correspond to the four TWAMP logical entities
        Control-Client, Server, Session-Sender, and Session-Reflector
        as illustrated in Fig. 1 of draft-ietf-ippm-twamp-yang.";

    container client {
      if-feature control-client;
      presence "Enables TWAMP Control-Client functionality.";
      description
        "Configuration of the TWAMP Control-Client logical entity.";

      leaf admin-state {
        type boolean;
        mandatory true;
```

```
        description
          "Indicates whether the device is allowed to operate as a
          TWAMP Control-Client.";
      }

      list mode-preference-chain {
        key priority;
        unique mode;
        leaf priority {
          type uint16;
          description
            "Indicates the Control-Client Mode preference priority
            expressed as a 16-bit unsigned integer, where zero is the
            highest priority and subsequent values monotonically
            increasing.";
        }
        leaf mode {
          type twamp-modes;
          description
            "The supported TWAMP Mode matching the corresponding
            priority.";

        }
        description
            "Indicates the Control-Client preferred order of use of
            the supported TWAMP Modes.

            Depending on the Modes available in the TWAMP Server
            Greeting message (see Fig. 2 of RFC 7717), the
            this Control-Client MUST choose the highest priority Mode
            from the configured mode-preference-chain list.";
      }

      uses key-management;

      list ctrl-connection {
        key name;
        description
          "List of TWAMP Control-Client control connections.
          Each item in the list describes a control connection
          that will be initiated by this Control-Client";

        leaf name {
          type string;
          description
            "A unique name used as a key to identify this individual
            TWAMP-Control connection on the Control-Client device.";
        }
```

```
        leaf client-ip {
          type inet:ip-address;
          description
            "The IP address of the local Control-Client device,
            to be placed in the source IP address field of the
            IP header in TWAMP-Control (TCP) packets belonging
            to this control connection. If not configured, the
            device SHALL choose its own source IP address.";
        }
        leaf server-ip {
          type inet:ip-address;
          mandatory true;
          description
            "The IP address of the remote Server device, which the
            TWAMP-Control connection will be initiated to.";
        }

        leaf server-tcp-port {
          type inet:port-number;
          default 862;
          description
            "This parameter defines the TCP port number that is
            to be used by this outgoing TWAMP-Control connection.
            Typically, this is the well-known TWAMP-Control
            port number (862) as per RFC 5357 However, there are known
            realizations of TWAMP in the field that were implemented
            before this well-known port number was allocated. These
            early implementations allowed the port number to be
            configured. This parameter is therefore provided for
            backward compatibility reasons.";
        }

        leaf control-packet-dscp {
          type inet:dscp;
          default 0;
          description
            "The DSCP value to be placed in the IP header of
            TWAMP-Control (TCP) packets generated by this
            Control-Client.";
        }

        leaf key-id {
          type string {
            length 1..80;
          }
          description
           "Indicates the KeyID value selected for this
           TWAMP-Control connection.";
```

```
        }

        uses max-count;

        leaf client-tcp-port {
          type inet:port-number;
          config false;
          description
            "Indicates the source TCP port number used in the
            TWAMP-Control packets belonging to this control
            connection.";
        }

        leaf server-start-time {
          type uint64;
          config false;
          description
            "Indicates the Start-Time advertized by the Server in the
            Server-Start message (RFC 4656, Section 3.1),
            representing the time when the current
            instantiation of the Server started operating.
            The timestamp format follows RFC 1305
            according to Section 4.1.2 of RFC 4656.";
        }

        leaf state {
          type control-client-connection-state;
          config false;
          description
            "Indicates the current state of the TWAMP-Control
            connection state.";
        }

        leaf selected-mode {
          type twamp-modes;
          config false;
          description
            "The TWAMP Mode that the Control-Client has chosen for
            this control connection as set in the Mode field of the
            Set-Up-Response message (RFC 4656, Section 3.1).";
        }

        leaf token {
            type binary {
            length 64;
          }
          config false;
          description
```

```
            "This parameter holds the 64 octets containing the
            concatenation of a 16-octet Challenge, a 16-octet AES
            Session-key used for encryption, and a 32-octet
            HMAC-SHA1 Session-key used for authentication; see also
            the last paragraph of Section 6 in RFC 4656.

            If the Mode defined in RFC 7717 is selected (selected-mode),
            Token is limited to 16 octets.";
          reference
            "RFC 4086: Randomness Requirements for Security

             RFC 7717: IKEv2-Derived Shared Secret Key for the One-Way
             Active Measurement Protocol (OWAMP) and Two-Way Active
             Measurement Protocol (TWAMP)";
        }

        leaf client-iv {
          type binary {
            length 16;
          }
          config false;
          description
            "Indicates the Control-Client Initialization Vector
            (Client-IV), that is generated randomly by the
            Control-Client. As per RFC 4656:

              Client-IV merely needs to be unique (i.e., it MUST
              never be repeated for different sessions using the
              same secret key; a simple way to achieve that without
              the use of cumbersome state is to generate the
              Client-IV values using a cryptographically secure
              pseudo-random number source.

            If the Mode defined in RFC 7717 is selected (selected-mode),
            Client-IV is limited to 12 octets.";
          reference
            "RFC 4656: A One-way Active Measurement Protocol (OWAMP)

             RFC 7717: IKEv2-Derived Shared Secret Key for the One-Way
             Active Measurement Protocol (OWAMP) and Two-Way Active
             Measurement Protocol (TWAMP)";        }

        list test-session-request {
          key name;
          description
            "Information associated with the Control-Client
            for this test session";
```

```
          leaf name {
            type string;
            description
              "A unique name to be used for identification of
              this TWAMP-Test session on the Control-Client.";
          }

          leaf sender-ip {
            type inet:ip-address;
            description
              "The IP address of the Session-Sender device,
              which is to be placed in the source IP address
              field of the IP header in TWAMP-Test (UDP) packets
              belonging to this test session. This value will be
              used to populate the sender address field of the
              Request-TW-Session message.

              If not configured, the device SHALL choose its own
              source IP address.";
          }

          leaf sender-udp-port {
            type union {
              type dynamic-port-number;
              type enumeration {
                enum autoallocate {
                  description
                    "Indicates that the Contol-Client will
                    auto-allocate the TWAMP-Test (UDP) port number
                    from the dynamic port range.";
                }
              }
            }
            default autoallocate;
            description
              "The UDP port number that is to be used by
              the Session-Sender for this TWAMP-Test session.
              The number is restricted to the dynamic port range.

              By default the Control-Client SHALL auto-allocate a
              UDP port number for this TWAMP-Test session.

              The configured (or auto-allocated) value is advertized
              in the Sender Port field of the Request-TW-session
              message (see Section 3.5 of RFC 5357). Note that
              in the scenario where a device auto-allocates a UDP
              port number for a session, and the repeat parameter
              for that session indicates that it should be
```

```
            repeated, the device is free to auto-allocate a
            different UDP port number when it negotiates the
            next (repeated) iteration of this session.";
          }

          leaf reflector-ip {
            type inet:ip-address;
            mandatory true;
            description
              "The IP address belonging to the remote
              Session-Reflector device to which the TWAMP-Test
              session will be initiated. This value will be
              used to populate the receiver address field of
              the Request-TW-Session message.";
          }

          leaf reflector-udp-port {
            type dynamic-port-number;
            description
              "This parameter defines the UDP port number that
              will be used by the Session-Reflector for
              this TWAMP-Test session. The number is restricted
              to the dynamic port range and is to be placed in
              the Receiver Port field of the Request-TW-Session
              message.";
          }

          leaf timeout {
            type uint64;
            units seconds;
            default 2;
            description
                "The length of time (in seconds) that the
                Session-Reflector should continue to respond to
                packets belonging to this TWAMP-Test session after
                a Stop-Sessions TWAMP-Control message has been
                received (RFC 5357, Section 3.8).

                This value will be placed in the Timeout field of
                the Request-TW-Session message.";
          }

          leaf padding-length {
            type uint32 {
              range 64..4096;
            }
            description
                "The number of padding bytes to be added to the
```

```
            TWAMP-Test (UDP) packets generated by the
            Session-Sender.

            This value will be placed in the Padding Length
            field of the Request-TW-Session message
            (RFC 4656, Section 3.5).";
          }

          leaf test-packet-dscp {
            type inet:dscp;
            default 0;
            description
                "The DSCP value to be placed in the IP header
                of TWAMP-Test packets generated by the
                Session-Sender, and in the UDP header of the
                TWAMP-Test response packets generated by the
                Session-Reflector for this test session.

                This value will be placed in the Type-P Descriptor
                field of the Request-TW-Session message (RFC 5357).";
          }

          leaf start-time {
            type uint64;
            default 0;
            description
                "Time when the session is to be started
                (but not before the TWAMP Start-Sessions command
                is issued; see Section 3.4 of RFC 5357).

                The start-time value is placed in the Start Time
                field of the Request-TW-Session message.

                The timestamp format follows RFC 1305 as per
                Section 3.5 of RFC 4656.

                The default value of 0 indicates that the session
                will be started as soon as the Start-Sessions message
                is received.";
          }

          leaf repeat {
            type union {
              type uint32 {
                range 0..4294967294;
              }
              type enumeration {
                enum forever {
```

```
                description
                  "Indicates that the test session SHALL be
                  repeated *forever* using the information in
                  repeat-interval parameter, and SHALL NOT
                  decrement the value.";
              }
            }
          }
          default 0;
          description
              "This value determines if the TWAMP-Test session must
              be repeated. When a test session has completed, the
              repeat parameter is checked.

              The default value of 0 indicates that the session
              MUST NOT be repeated.

              If the repeat value is 1 through 4,294,967,294
              then the test session SHALL be repeated using the
              information in repeat-interval parameter, and the
              parent TWAMP-Control connection for this test
              session is restarted to negotiate a new instance
              of this TWAMP-Test session. The implementation
              MUST decrement the value of repeat after
              determining a repeated session is expected.";
        }

        leaf repeat-interval  {
          when "../repeat!='0'" {
            description
              "This parameter determines the timing of repeated
              TWAMP-Test sessions when repeat is more than 0.

              When the value of repeat-interval is 0, the
              negotiation of a new test session SHALL begin
              immediately after the previous test session
              completes. Otherwise, the Control-Client will
              wait for the number of seconds specified in the
              repeat-interval parameter before negotiating the
              new instance of this TWAMP-Test session.";
          }
          type uint32;
          units seconds;
          default 0;
          description "Repeat interval (in seconds).";
        }

        list pm-reg-list {
```

```
            key pm-index;
            leaf pm-index {
              type uint16;
              description
                "Numerical index value of a Registered Metric
                in the Performance Metric Registry
                (see ietf-ippm-metric-registry). Output statistics
                are specified in the corresponding Registry entry.";
            }
            description
                "A list of one or more Performance Metric Registry
                Index values, which communicate packet stream
                characteristics along with one or more metrics
                to be measured.

                All members of the pm-reg-list MUST have the same
                stream characteristics, such that they combine
                to specify all metrics that shall be measured on
                a single stream.";
            reference
              "ietf-ippm-metric-registry:
                Registry for Performance Metrics";
          }
          leaf state {
            type test-session-state;
            config false;
            description
                "Indicates the TWAMP-Test session state (accepted or
                indication of an error); see Section 3.5 of
                RFC 5357.";
          }
          leaf sid {
            type string;
            config false;
            description
                "The SID allocated by the Server for this TWAMP-Test
                session, and communicated back to the Control-Client
                in the SID field of the Accept-Session message;
                see Section 4.3 of RFC 6038.";
          }
        }
      }
    }

    container server {
      if-feature server;
      presence "Enables TWAMP Server functionality.";
      description "Configuration of the TWAMP Server logical entity.";
```

```
      leaf admin-state {
        type boolean;
        mandatory true;
        description
            "Indicates whether the device is allowed to operate
            as a TWAMP Server.";
      }

      leaf server-tcp-port {
        type inet:port-number;
        default 862;
        description
            "This parameter defines the well known TCP port number
            that is used by TWAMP-Control. The Server will listen
            on this port number for incoming TWAMP-Control
            connections. Although this is defined as a fixed value
            (862) in RFC 5357, there are several realizations of
            TWAMP in the field that were implemented before this
            well-known port number was allocated. These early
            implementations allowed the port number to be
            configured. This parameter is therefore provided for
            backward compatibility reasons.";
      }

      leaf servwait {
        type uint32 {
          range 1..604800;
        }
        units seconds;
        default 900;
        description
          "TWAMP-Control (TCP) session timeout, in seconds.
          According to Section 3.1 of RFC 5357,

            Server MAY discontinue any established control
            connection when no packet associated with that
            connection has been received within SERVWAIT seconds.";
      }

      leaf control-packet-dscp {
        type inet:dscp;
        description
            "The DSCP value to be placed in the IP header of
            TWAMP-Control (TCP) packets generated by the Server.

            Section 3.1 of  RFC 5357 specifies that the server
            SHOULD use the DSCP value from the Control-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 RFC 5357 in an
            OS-portable version of TWAMP.

            The default behavior if this item is not set is to use
            the DSCP value from the Control-Client's TCP SYN,
            as per Section 3.1 of RFC 5357.";
        }

        uses count;

        uses max-count;

        leaf modes {
          type twamp-modes;
          description
              "The bit mask of TWAMP Modes this Server instance
              is willing to support; see IANA TWAMP Modes Registry.";
        }

        uses key-management;

        list ctrl-connection {
          key "client-ip client-tcp-port server-ip server-tcp-port";
          config false;
          description
              "List of all incoming TWAMP-Control (TCP) connections.";

          leaf client-ip {
            type inet:ip-address;
            description
              "The IP address on the remote Control-Client device,
              which is the source IP address used in the
              TWAMP-Control (TCP) packets belonging to this control
              connection.";
          }

          leaf client-tcp-port {
            type inet:port-number;
            description
              "The source TCP port number used in the TWAMP-Control
              (TCP) packets belonging to this control connection.";
          }

          leaf server-ip {
```

```
          type inet:ip-address;
          description
            "The IP address of the local Server device, which is
            the destination IP address used in the
            TWAMP-Control (TCP) packets belonging to this control
            connection.";
        }

        leaf server-tcp-port {
          type inet:port-number;
          description
            "The destination TCP port number used in the
            TWAMP-Control (TCP) packets belonging to this
            control connection. This will usually be the
            same value as the server-tcp-port configured
            under twamp/server. However, in the event that
            the user re-configured server/server-tcp-port
            after this control connection was initiated, this
            value will indicate the server-tcp-port that is
            actually in use for this control connection.";
        }

        leaf state {
          type server-ctrl-connection-state;
          description
            "Indicates the Server TWAMP-Control connection state.";
        }

        leaf control-packet-dscp {
          type inet:dscp;
          description
            "The DSCP value used in the IP header of the
            TWAMP-Control (TCP) packets sent by the Server
            for this control connection. This will usually
            be the same value as is configured in the
            control-packet-dscp parameter under the twamp/server
            container.  However, in the event that the user
            re-configures server/dscp after this control
            connection is already in progress, this read-only
            value will show the actual dscp value in use by this
            TWAMP-Control connection.";
        }

        leaf selected-mode {
          type twamp-modes;
          description
            "The Mode that was chosen for this TWAMP-Control
            connection as set in the Mode field of the
```

```
          Set-Up-Response message.";
        }

        leaf key-id {
          type string {
            length 1..80;
          }
          description
            "The KeyID value that is in use by this TWAMP-Control
            connection as selected by Control-Client.";
        }

        uses count {
          description
            "The count value that is in use by this TWAMP-Control
            connection. This will usually be the same value
            as is configured under twamp/server. However, in the
            event that the user re-configured server/count
            after this control connection is already in progress,
            this read-only value will show the actual count that
            is in use for this TWAMP-Control connection.";
        }

        uses max-count {
          description
            "This read-only value indicates the actual max-count in use
            for this control connection. Usually this would be the
            same value as configured under twamp/server.";
        }

        leaf salt {
          type binary {
            length 16;
          }
          description
            "A parameter used in deriving a key from a
            shared secret as described in Section 3.1 of RFC 4656.
            It is communicated to the Control-Client as part of
            the Server Greeting message.";
        }

        leaf server-iv {
          type binary {
            length 16;
          }
          description
            "The Server Initialization Vector
            (IV) generated randomly by the Server.";
```

```
        }

        leaf challenge {
          type binary {
            length 16;
          }
          description
            "A random sequence of octets generated by the Server.
            As described in client/token, Challenge is used
            by the Control-Client to prove possession of a
            shared secret.";
        }
      }
    }

    container session-sender {
      if-feature session-sender;
      presence  "Enables TWAMP Session-Sender functionality.";
      description
        "Configuration of the TWAMP Session-Sender logical entity";
      leaf admin-state {
        type boolean;
        mandatory true;
        description
            "Indicates whether the device is allowed to operate
            as a TWAMP Session-Sender.";
      }

      list test-session{
        key name;
        description "List of TWAMP Session-Sender test sessions.";

        leaf name {
          type string;
          description
            "A unique name for this TWAMP-Test session to be used
            for identifying this test session by the Session-Sender
            logical entity.";
        }

        leaf ctrl-connection-name {
          type string;
          config false;
          description
            "The name of the parent TWAMP-Control connection that
            is responsible for negotiating this TWAMP-Test session.";
        }
```

```
        leaf fill-mode {
          type padding-fill-mode;
          default zero;
          description
            "Indicates whether the padding added to the
            TWAMP-Test (UDP) packets will contain pseudo-random
            numbers, or whether it should consist of all zeroes,
            as per Section 4.2.1 of RFC 5357.";
        }

        leaf number-of-packets {
          type uint32;
          mandatory true;
          description
            "The overall number of TWAMP-Test (UDP) packets to be
            transmitted by the Session-Sender for this test session.";
        }

        choice packet-distribution {
          description
            "Indicates the distribution to be used for transmitting
            the TWAMP-Test (UDP) packets.";
          case periodic {
            leaf periodic-interval {
              type decimal64 {
                fraction-digits 5;
              }
              units seconds;
              mandatory true;
              description
                "Indicates the time to wait (in seconds) between the
                first bits of TWAMP-Test (UDP) packet transmissions for
                this test session";
              reference
                "RFC 3432: Network performance measurement
                           with periodic streams";
            }
          }
          case poisson {
            leaf lambda {
              type decimal64 {
                fraction-digits 5;
              }
              units seconds;
              mandatory true;
              description
                "Indicates the average time interval (in seconds)
                between packets in the Poisson distribution.
```

```
                The packet is calculated using the reciprocal of lambda
                and the TWAMP-Test packet size (which depends on the
                selected Mode and the packet padding).";
              reference
                "RFC 2330: Framework for IP Performance Metrics";
            }
            leaf max-interval {
              type decimal64 {
                fraction-digits 5;
              }
              units seconds;
              description
                "Indicates the maximum time (in seconds)
                between packet transmissions.";
              reference
                "RFC 7312: Advanced Stream and Sampling Framework
                           for IP Performance Metrics (IPPM)";
            }
          }
        }

        leaf state {
          type sender-session-state;
          config false;
          description
            "Indicates the Session-Sender test session state.";
        }

        uses maintenance-statistics;
      }
    }

    container session-reflector {
      if-feature session-reflector;
      presence "Enables TWAMP Session-Reflector functionality.";
      description
        "Configuration of the TWAMP Session-Reflector logical entity";

      leaf admin-state {
        type boolean;
        mandatory true;
        description
          "Indicates whether the device is allowed to operate
          as a TWAMP Session-Reflector.";
      }

      leaf refwait {
        type uint32 {
```

```
          range 1..604800;
        }
        units seconds;
        default 900;
        description
          "The Session-Reflector MAY discontinue any session that has
          been started when no packet associated with that session has
          been received for REFWAIT seconds. As per Section 3.1 of
          RFC 5357, this timeout allows a Session-Reflector to free up
          resources in case of failure.";
      }

      list test-session {
        key
          "sender-ip sender-udp-port
          reflector-ip reflector-udp-port";
        config false;
        description "TWAMP Session-Reflectortest sessions.";

        leaf sid {
          type string;
          description
            "An auto-allocated identifier for this TWAMP-Test
            session that is unique within the context of this
            Server/Session-Reflector device only. This value
            is communicated to the Control-Client that
            requested the test session in the SID field of the
            Accept-Session message.";
        }

        leaf sender-ip {
          type inet:ip-address;
          description
            "The IP address on the remote device, which is the
            source IP address used in the TWAMP-Test (UDP) packets
            belonging to this test session.";
        }

        leaf sender-udp-port {
          type dynamic-port-number;
          description
            "The source UDP port used in the TWAMP-Test packets
            belonging to this test session.";
        }

        leaf reflector-ip {
          type inet:ip-address;
          description
```

```
            "The IP address of the local Session-Reflector
            device, which is the destination IP address used
            in the TWAMP-Test (UDP) packets belonging to this test
            session.";
          }

          leaf reflector-udp-port {
            type dynamic-port-number;
            description
              "The destination UDP port number used in the
              TWAMP-Test (UDP) test packets belonging to this
              test session.";
          }

          leaf parent-connection-client-ip {
            type inet:ip-address;
            description
              "The IP address on the Control-Client device, which
              is the source IP address used in the TWAMP-Control
              (TCP) packets belonging to the parent control
              connection that negotiated this test session.";
          }

          leaf parent-connection-client-tcp-port {
            type inet:port-number;
            description
              "The source TCP port number used in the TWAMP-Control
              (TCP) packets belonging to the parent control connection
              that negotiated this test session.";
          }

          leaf parent-connection-server-ip {
            type inet:ip-address;
            description
              "The IP address of the Server device, which is the
              destination IP address used in the TWAMP-Control
              (TCP) packets belonging to the parent control
              connection that negotiated this test session.";
          }

          leaf parent-connection-server-tcp-port {
            type inet:port-number;
            description
              "The destination TCP port number used in the TWAMP-Control
              (TCP) packets belonging to the parent control connection
              that negotiated this test session.";
          }
```

```
        leaf test-packet-dscp {
          type inet:dscp;
          description
            "The DSCP value present in the IP header of
            TWAMP-Test (UDP) packets belonging to this session.";
        }

        uses maintenance-statistics;
      }
    }
  }
}
```

<CODE ENDS>

6. Data Model Examples

This section presents a simple but complete example of configuring all four entities in Figure 1, based on the YANG module specified in Section 5. The example is illustrative in nature, but aims to be self-contained, i.e. were it to be executed in a real TWAMP implementation it would lead to a correctly configured test session. For completeness, examples are provided for both IPv4 and IPv6.

A more elaborated example, which also includes authentication parameters, is provided in Appendix A.

6.1. Control-Client

Figure 8 shows a configuration example for a Control-Client with client/admin-state enabled. In a real implementation following Figure 2 this would permit the initiation of TWAMP-Control connections and TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <client>
         <admin-state>true</admin-state>
      </client>
   </twamp>
</config>
```

Figure 8: XML instance enabling Control-Client operation.

The following example shows a Control-Client with two instances of client/ctrl-connection, one called "RouterA" and another called

"RouterB". Each TWAMP-Control connection is to a different Server. The control connection named "RouterA" has two test session requests. The TWAMP-Control connection named "RouterB" has no TWAMP-Test session requests.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <client>
         <admin-state>true</admin-state>
         <ctrl-connection>
            <name>RouterA</name>
            <client-ip>203.0.113.1</client-ip>
            <server-ip>203.0.113.2</server-ip>
            <test-session-request>
               <name>Test1</name>
               <sender-ip>203.0.113.3</sender-ip>
               <sender-udp-port>54001</sender-udp-port>
               <reflector-ip>203.0.113.4</reflector-ip>
               <reflector-udp-port>50001</reflector-udp-port>
               <start-time>0</start-time>
            </test-session-request>
            <test-session-request>
               <name>Test2</name>
               <sender-ip>203.0.113.1</sender-ip>
               <sender-udp-port>54001</sender-udp-port>
               <reflector-ip>203.0.113.2</reflector-ip>
               <reflector-udp-port>50001</reflector-udp-port>
               <start-time>0</start-time>
            </test-session-request>
         </ctrl-connection>
         <ctrl-connection>
            <name>RouterB</name>
            <client-ip>203.0.113.1</client-ip>
            <server-ip>203.0.113.3</server-ip>
         </ctrl-connection>
      </client>
   </twamp>
</config>

<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <client>
         <admin-state>true</admin-state>
         <ctrl-connection>
            <name>RouterA</name>
            <client-ip>2001:DB8:203:0:113::1</client-ip>
```

```
               <server-ip>2001:DB8:203:0:113::2</server-ip>
               <test-session-request>
                  <name>Test1</name>
                  <sender-ip>2001:DB8:203:1:113::3</sender-ip>
                  <sender-udp-port>54000</sender-udp-port>
                  <reflector-ip>2001:DB8:203:1:113::4</reflector-ip>
                  <reflector-udp-port>55000</reflector-udp-port>
                  <start-time>0</start-time>
               </test-session-request>
               <test-session-request>
                  <name>Test2</name>
                  <sender-ip>2001:DB8:203:0:113::1</sender-ip>
                  <sender-udp-port>54001</sender-udp-port>
                  <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
                  <reflector-udp-port>55001</reflector-udp-port>
                  <start-time>0</start-time>
               </test-session-request>
            </ctrl-connection>
            <ctrl-connection>
               <name>RouterB</name>
               <client-ip>2001:DB8:203:0:113::1</client-ip>
               <server-ip>2001:DB8:203:0:113::3</server-ip>
            </ctrl-connection>
         </client>
      </twamp>
   </config>
```

6.2. Server

Figure 9 shows a configuration example for a Server with server/admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Control connections and TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <server>
         <admin-state>true</admin-state>
      </server>
   </twamp>
</config>
```

Figure 9: XML instance enabling Server operation.

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

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <server>
         <admin-state>true</admin-state>
         <ctrl-connection>
            <client-ip>203.0.113.1</client-ip>
            <client-tcp-port>16341</client-tcp-port>
            <server-ip>203.0.113.2</server-ip>
            <server-tcp-port>862</server-tcp-port>
            <state>
               active
            </state>
         </ctrl-connection>
      </server>
   </twamp>
</data>

<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <server>
         <admin-state>true</admin-state>
         <ctrl-connection>
            <client-ip>2001:DB8:203:0:113::1</client-ip>
            <client-tcp-port>16341</client-tcp-port>
            <server-ip>2001:DB8:203:0:113::2</server-ip>
            <server-tcp-port>862</server-tcp-port>
            <state>
               active
            </state>
         </ctrl-connection>
      </server>
   </twamp>
</data>
```

6.3. Session-Sender

Figure 10 shows a configuration example for a Session-Sender with session-sender/admin-state enabled, which permits a device following Figure 2 to initiate TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <session-sender>
         <admin-state>true</admin-state>
      </session-sender>
   </twamp>
</config>
```

Figure 10: XML instance enabling Session-Sender operation.

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

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
     <session-sender>
        <admin-state>true</admin-state>
        <test-session>
           <name>Test1</name>
           <ctrl-connection-name>RouterA</ctrl-connection-name>
           <number-of-packets>900</number-of-packets>
           <periodic-interval>1</periodic-interval>
           <state>setup</state>
        </test-session>
        <test-session>
           <name>Test2</name>
           <ctrl-connection-name>
             RouterA
           </ctrl-connection-name>
           <number-of-packets>900</number-of-packets>
           <lambda>1</lambda>
           <max-interval>2</max-interval>
           <state>setup</state>
        </test-session>
     </session-sender>
  </twamp>
</data>
```

6.4. Session-Reflector

This configuration example shows a Session-Reflector with session-reflector/admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <session-reflector>
         <admin-state>true</admin-state>
      </session-reflector>
   </twamp>
</config>
```

Figure 11: XML instance enabling Session-Reflector operation.

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

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <session-reflector>
          <admin-state>true</admin-state>
          <test-session>
            <sender-ip>203.0.113.3</sender-ip>
            <sender-udp-port>54000</sender-udp-port>
            <reflector-ip>203.0.113.4</reflector-ip>
            <reflector-udp-port>50001</reflector-udp-port>
            <sid>1232</sid>
            <parent-connection-client-ip>
               203.0.113.1
            </parent-connection-client-ip>
            <parent-connection-client-tcp-port>
               16341
            </parent-connection-client-tcp-port>
            <parent-connection-server-ip>
               203.0.113.2
            </parent-connection-server-ip>
            <parent-connection-server-tcp-port>
               862
             </parent-connection-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>
         </test-session>
         <test-session>
            <sender-ip>203.0.113.1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>192.68.0.2</reflector-ip>
            <reflector-udp-port>50001</reflector-udp-port>
            <sid>178943</sid>
```

```
              <parent-connection-client-ip>
                 203.0.113.1
              </parent-connection-client-ip>
              <parent-connection-client-tcp-port>
                 16341
               </parent-connection-client-tcp-port>
              <parent-connection-server-ip>
                 203.0.113.2
              </parent-connection-server-ip>
              <parent-connection-server-tcp-port>
                862
              </parent-connection-server-tcp-port>
              <sent-packets>21</sent-packets>
              <rcv-packets>21</rcv-packets>
              <last-sent-seq>20</last-sent-seq>
              <last-rcv-seq>20</last-rcv-seq>
           </test-session>
        </session-reflector>
     </twamp>
  </data>

  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
        <session-reflector>
            <admin-state>true</admin-state>
            <test-session>
              <sender-ip>203.0.113.3</sender-ip>
              <sender-udp-port>54000</sender-udp-port>
              <reflector-ip>203.0.113.4</reflector-ip>
              <reflector-udp-port>54001</reflector-udp-port>
              <sid>1232</sid>
              <parent-connection-client-ip>
                  203.0.113.1
              </parent-connection-client-ip>
              <parent-connection-client-tcp-port>
                 16341
              </parent-connection-client-tcp-port>
              <parent-connection-server-ip>
                  203.0.113.2
              </parent-connection-server-ip>
              <parent-connection-server-tcp-port>
                 862
               </parent-connection-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>
```

```
            </test-session>
            <test-session>
               <sender-ip>203.0.113.1</sender-ip>
               <sender-udp-port>54001</sender-udp-port>
               <reflector-ip>192.68.0.2</reflector-ip>
               <reflector-udp-port>55001</reflector-udp-port>
               <sid>178943</sid>
               <parent-connection-client-ip>
                   203.0.113.1
               </parent-connection-client-ip>
               <parent-connection-client-tcp-port>
                  16341
                </parent-connection-client-tcp-port>
               <parent-connection-server-ip>
                   203.0.113.2
               </parent-connection-server-ip>
               <parent-connection-server-tcp-port>
                 862
               </parent-connection-server-tcp-port>
               <sent-packets>21</sent-packets>
               <rcv-packets>21</rcv-packets>
               <last-sent-seq>20</last-sent-seq>
               <last-rcv-seq>20</last-rcv-seq>
            </test-session>
         </session-reflector>
      </twamp>
   </data>
```

7. Security Considerations

The YANG module defined in Section 5 is designed to be accessed, among other protocols, via NETCONF [RFC6241]. Protocols like NETCONF use a secure transport layer like SSH that is mandatory to implement. The NETCONF Access Control Module (NACM) [RFC6536] provides the means to restrict access for particular users to a pre-configured set of NETCONF protocol operations and attributes.

There are a number of nodes defined in this YANG module which are writeable. These data nodes may be considered sensitive and vulnerable to attacks in some network environments. Ability to write into these nodes without proper protection can have a negative effect on the devices that support this feature.

Examples of nodes that are particularly vulnerable include several timeout values put in the protocol to protect against sessions that are not active but are consuming resources.

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 Fred Baker, Kevin D'Souza, Gregory Mirsky, Brian Trammell, Robert Sherman, and Marius Georgescu for their thorough and constructive reviews, comments and text suggestions.

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

Jan Lindblad and Ladislav Lhokta did thorough reviews of the YANG module and the examples in Appendix A.

Kostas Pentikousis was partially supported by FP7 UNIFY (http://fp7-unify.eu), a research project partially funded by the European Community under the Seventh Framework Program (grant agreement no. 619609). The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

10. Contributors

Lianshu Zheng, Huawei Technologies, China

Email: vero.zheng@huawei.com

11. References

11.1. 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>.

[RFC7717] Pentikousis, K., Ed., Zhang, E., and Y. Cui, "IKEv2-Derived Shared Secret Key for the One-Way Active Measurement Protocol (OWAMP) and Two-Way Active Measurement Protocol (TWAMP)", RFC 7717, DOI 10.17487/RFC7717, December 2015, <http://www.rfc-editor.org/info/rfc7717>.

11.2. 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-10 (work in progress), November 2016.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-18 (work in progress), October 2016.

[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-04 (work in progress), July 2016.

[I-D.unify-nfvrg-devops]
Meirosu, C., Manzalini, A., Steinert, R., Marchetto, G., Pentikousis, K., Wright, S., Lynch, P., and W. John, "DevOps for Software-Defined Telecom Infrastructures", draft-unify-nfvrg-devops-06 (work in progress), July 2016.

[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>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[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

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <client>
         <admin-state>true</admin-state>
         <mode-preference-chain>
            <priority>0</priority>
            <mode>authenticated</mode>
         </mode-preference-chain>
         <mode-preference-chain>
            <priority>1</priority>
            <mode>unauthenticated</mode>
         </mode-preference-chain>
         <key-chain>
            <key-id>KeyClient1ToRouterA</key-id>
            <secret-key>secret1</secret-key>
         </key-chain>
         <key-chain>
            <key-id>KeyForRouterB</key-id>
            <secret-key>secret2</secret-key>
         </key-chain>
         <ctrl-connection>
            <name>RouterA</name>
            <client-ip>203.0.113.1</client-ip>
```

```
               <server-ip>203.0.113.2</server-ip>
               <dscp>32</dscp>
               <key-id>KeyClient1ToRouterA</key-id>
               <test-session-request>
                  <name>Test1</name>
                  <sender-ip>203.0.113.3</sender-ip>
                  <sender-udp-port>54000</sender-udp-port>
                  <reflector-ip>203.0.113.4</reflector-ip>
                  <reflector-udp-port>55000</reflector-udp-port>
                  <padding-length>64</padding-length>
                  <start-time>0</start-time>
                  <state>ok</state>
                  <sid>1232</sid>
               </test-session-request>
               <test-session-request>
                  <name>Test2</name>
                  <sender-ip>203.0.113.1</sender-ip>
                  <sender-udp-port>54001</sender-udp-port>
                  <reflector-ip>203.0.113.2</reflector-ip>
                  <reflector-udp-port>55001</reflector-udp-port>
                  <padding-length>128</padding-length>
                  <start-time>0</start-time>
                  <state>ok</state>
                  <sid>178943</sid>
               </test-session-request>
            </ctrl-connection>
         </client>
      </twamp>
   </data>

   <?xml version="1.0" encoding="utf-8"?>
   <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
         <client>
            <admin-state>true</admin-state>
            <mode-preference-chain>
               <priority>0</priority>
               <mode>authenticated</mode>
            </mode-preference-chain>
            <mode-preference-chain>
               <priority>1</priority>
               <mode>unauthenticated</mode>
            </mode-preference-chain>
            <key-chain>
               <key-id>KeyClient1ToRouterA</key-id>
               <secret-key>secret1</secret-key>
            </key-chain>
            <key-chain>
```

```
               <key-id>KeyForRouterB</key-id>
               <secret-key>secret2</secret-key>
            </key-chain>
            <ctrl-connection>
               <name>RouterA</name>
               <client-ip>2001:DB8:203:0:113::1</client-ip>
               <server-ip>2001:DB8:203:0:113::2</server-ip>
               <dscp>32</dscp>
               <key-id>KeyClient1ToRouterA</key-id>
               <test-session-request>
                  <name>Test1</name>
                  <sender-ip>2001:DB8:10:1:1::1</sender-ip>
                  <sender-udp-port>54000</sender-udp-port>
                  <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
                  <reflector-udp-port>55000</reflector-udp-port>
                  <padding-length>64</padding-length>
                  <start-time>0</start-time>
                  <state>ok</state>
                  <sid>1232</sid>
               </test-session-request>
               <test-session-request>
                  <name>Test2</name>
                  <sender-ip>2001:DB8:203:0:113::1</sender-ip>
                  <sender-udp-port>54001</sender-udp-port>
                  <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
                  <reflector-udp-port>55001</reflector-udp-port>
                  <padding-length>128</padding-length>
                  <start-time>0</start-time>
                  <state>ok</state>
                  <sid>178943</sid>
               </test-session-request>
            </ctrl-connection>
         </client>
      </twamp>
   </data>
```

A.2. Server

```
   <?xml version="1.0" encoding="utf-8"?>
   <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
         <server>
            <admin-state>true</admin-state>
            <servwait>1800</servwait>
            <dscp>32</dscp>
            <modes>authenticated unauthenticated</modes>
            <count>1024</count>
            <key-chain>
```

```
               <key-id>KeyClient1ToRouterA</key-id>
               <secret-key>secret1</secret-key>
            </key-chain>
            <key-chain>
               <key-id>KeyClient10ToRouterA</key-id>
               <secret-key>secret10</secret-key>
            </key-chain>
            <ctrl-connection>
               <client-ip>203.0.113.1</client-ip>
               <client-tcp-port>16341</client-tcp-port>
               <server-ip>203.0.113.2</server-ip>
               <server-tcp-port>862</server-tcp-port>
               <state>
                  active
                </state>
               <dscp>32</dscp>
               <selected-mode>unauthenticated</selected-mode>
               <key-id>KeyClient1ToRouterA</key-id>
               <count>1024</count>
            </ctrl-connection>
         </server>
      </twamp>
   </data>

   <?xml version="1.0" encoding="utf-8"?>
   <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
         <server>
            <admin-state>true</admin-state>
            <servwait>1800</servwait>
            <dscp>32</dscp>
            <modes>authenticated unauthenticated</modes>
            <count>1024</count>
            <key-chain>
               <key-id>KeyClient1ToRouterA</key-id>
               <secret-key>secret1</secret-key>
            </key-chain>
            <key-chain>
               <key-id>KeyClient10ToRouterA</key-id>
               <secret-key>secret10</secret-key>
            </key-chain>
            <ctrl-connection>
               <client-ip>2001:DB8:203:0:113::1</client-ip>
               <client-tcp-port>16341</client-tcp-port>
               <server-ip>2001:DB8:203:0:113::2</server-ip>
               <server-tcp-port>862</server-tcp-port>
               <state>
                  active
```

```
                </state>
               <dscp>32</dscp>
               <selected-mode>unauthenticated</selected-mode>
               <key-id>KeyClient1ToRouterA</key-id>
               <count>1024</count>
            </ctrl-connection>
         </server>
      </twamp>
   </data>
```

A.3. Session-Sender

```
   <?xml version="1.0" encoding="utf-8"?>
   <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
        <session-sender>
           <admin-state>true</admin-state>
           <test-session>
              <name>Test1</name>
              <ctrl-connection-name>RouterA</ctrl-connection-name>
              <fill-mode>zero</fill-mode>
              <number-of-packets>900</number-of-packets>
              <periodic-interval>1</periodic-interval>
              <state>setup</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>
           </test-session>
           <test-session>
              <name>Test2</name>
              <ctrl-connection-name>
                RouterA
              </ctrl-connection-name>
              <fill-mode>random</fill-mode>
              <number-of-packets>900</number-of-packets>
              <lambda>1</lambda>
              <max-interval>2</max-interval>
              <state>setup</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>
           </test-session>
        </session-sender>
     </twamp>
   </data>
```

A.4. Session-Reflector

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <session-reflector>
         <admin-state>
            true
         </admin-state>
         <test-session>
            <sender-ip>203.0.113.3</sender-ip>
            <sender-udp-port>54000</sender-udp-port>
            <reflector-ip>203.0.113.4</reflector-ip>
            <reflector-udp-port>55000</reflector-udp-port>
            <sid>1232</sid>
            <parent-connection-client-ip>
               203.0.113.1
            </parent-connection-client-ip>
            <parent-connection-client-tcp-port>
               16341
            </parent-connection-client-tcp-port>
            <parent-connection-server-ip>
               203.0.113.2
            </parent-connection-server-ip>
            <parent-connection-server-tcp-port>
               862
            </parent-connection-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>
         </test-session>
         <test-session>
            <sender-ip>203.0.113.1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>192.68.0.2</reflector-ip>
            <reflector-udp-port>55001</reflector-udp-port>
            <sid>178943</sid>
            <parent-connection-client-ip>
               203.0.113.1
            </parent-connection-client-ip>
            <parent-connection-client-tcp-port>
               16341
            </parent-connection-client-tcp-port>
            <parent-connection-server-ip>
               203.0.113.2
            </parent-connection-server-ip>
```

```
               <parent-connection-server-tcp-port>
                  862
               </parent-connection-server-tcp-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>
            </test-session>
         </session-reflector>
      </twamp>
   </data>

   <?xml version="1.0" encoding="utf-8"?>
   <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
         <session-reflector>
            <admin-state>true</admin-state>
            <test-session>
               <sender-ip>2001:DB8:10:1:1::1</sender-ip>
               <sender-udp-port>54000</sender-udp-port>
               <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
               <reflector-udp-port>55000</reflector-udp-port>
               <sid>1232</sid>
               <parent-connection-client-ip>
                   2001:DB8:203:0:113::1
               </parent-connection-client-ip>
               <parent-connection-client-tcp-port>
                  16341
               </parent-connection-client-tcp-port>
               <parent-connection-server-ip>
                   2001:DB8:203:0:113::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>
            </test-session>
            <test-session>
               <sender-ip>2001:DB8:203:0:113::1</sender-ip>
               <sender-udp-port>54001</sender-udp-port>
               <reflector-ip>2001:DB8:192:68::2</reflector-ip>
               <reflector-udp-port>55001</reflector-udp-port>
               <sid>178943</sid>
```

```
               <parent-connection-client-ip>
                   2001:DB8:203:0:113::1
               </parent-connection-client-ip>
               <parent-connection-client-tcp-port>
                  16341
               </parent-connection-client-tcp-port>
               <parent-connection-server-ip>
                   2001:DB8:203:0:113::2
               </parent-connection-server-ip>
               <parent-connection-server-tcp-port>
                  862
               </parent-connection-server-tcp-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>
            </test-session>
         </session-reflector>
      </twamp>
   </data>
```

Appendix B. TWAMP Operational Commands

TWAMP operational commands could be performed programmatically or manually, e.g. using a command-line interface (CLI).

With respect to programmability, YANG can be used to define NETCONF Remote Procedure Calls (RPC), therefore it would be, in principle, possible to define TWAMP RPC operations for actions such as starting or stopping control connections or test sessions or groups of sessions; retrieving results; clearing stored results, and so on.

However, [RFC5357] does not attempt to describe such operational actions. Refer also to Section 2 and the unlabeled links in Figure 1. In actual deployments different TWAMP implementations may support different sets of operational commands, with different restrictions. Therefore, this document considers it the responsibility of the individual implementation to define its corresponding TWAMP operational commands data model.

Authors' Addresses

Ruth Civil
Ciena Corporation
307 Legget Drive
Kanata, ON K2K 3C8
Canada

Email: gcivil@ciena.com
URI: www.ciena.com

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

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com

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

Email: rrahman@cisco.com

Mahesh Jethanandani
Cisco Systems
3700 Cisco Way
San Jose, CA 95134
USA

Email: mjethanandani@gmail.com

Kostas Pentikousis (editor)
Travelping
Koernerstr. 7-10
Berlin 10785
Germany

Email: k.pentikousis@travelping.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 15, 2017

G. Mirsky
X. Min
ZTE Corp.
A. Pan
W. Luo
Ericsson

June 13, 2017

Two-Way Active Measurement Protocol (TWAMP) Light Data Model draft-mirsky-ippm-twamp-light-yang-09

Abstract

This document specifies the data model for implementations of Session-Sender and Session-Reflector for Two-Way Active Measurement Protocol (TWAMP) Light mode using YANG.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 15, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Two-Way Active Measurement Protocol (TWAMP) [RFC5357] can be used to measure performance parameters of IP networks such as latency, jitter, and packet loss by sending test packets and monitoring their experience in the network. The [RFC5357] defines two protocols, TWAMP Control and TWAMP Test, and a profile of TWAMP Test, TWAMP Light. The TWAMP Light is known to have many implementations though no common management framework being defined, thus leaving some aspects of test packet processing to interpretation. The goal of this document is to collect analyze these variations; describe common model while allowing for extensions in the future. This document defines such a TWAMP data model and specifies it formally using the YANG data modeling language [RFC6020].

1.1. Conventions used in this document

1.1.1. Requirements Language

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

2. Scope, Model, and Applicability

The scope of this document includes model of the TWAMP Light as defined in Appendix I of [RFC5357]. This mode of TWAMP Light will be referred in this document as Stateless. Another mode, where the Session-Reflector is aware of the state of the TWAMP test session and thus can independently count reflected test packets, referred as Stateful. This document benefits from earlier attempt to define TWAMP MIB in [I-D.elteto-ippm-twamp-mib] and from TWAMP YANG model defined in [I-D.ietf-ippm-twamp-yang].

Figure 1 updates TWAMP-Light reference model presented in Appendix I [RFC5357] for the scenario when instantiation of a TWAMP-Test session between Session-Sender and Session-Reflector controlled by communication between a Configuration Client as a manager and Configuration Servers as agents of the configuration session.

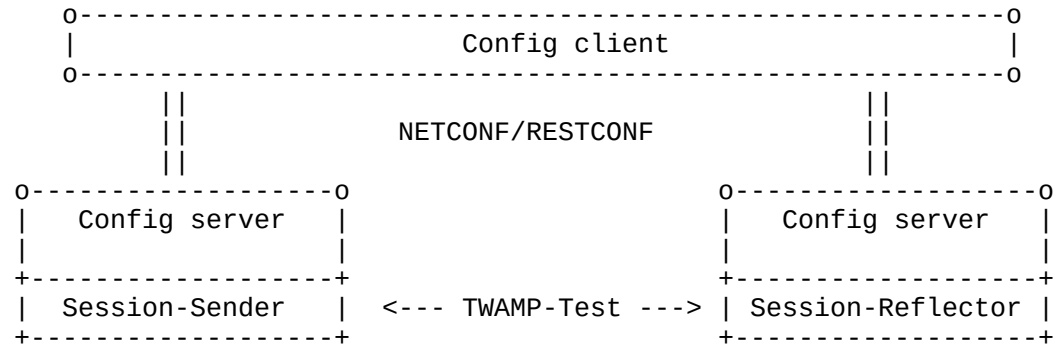


Figure 1: TWAMP Light Reference Model

2.1. Data Model Parameters

This section describes all the parameters of the the TWAMP-Light data model.

2.1.1. Session-Sender

The twamp-light-session-sender container holds items that are related to the configuration of the TWAMP-Light Session-Sender logical entity.

The twamp-light-session-sender-state container holds information about the state of the particular TWAMP-Light test session.

RPCs twamp-sender-start and twamp-sender-stop respectively start and stop the referenced by session-id TWAMP-Light test session.

2.1.1.1. Controls for Test Session and Preforrmance Metric Calculation

The data model supports several scenarios for Session-Sender to execute test sessions and calculate performance metrics:

The test mode in which the test packets are sent unbound in time at defined by the parameter 'interval' in the twamp-light-session-sender container frequency is referred as continuous mode. Performance metrics in the continuous mode are calculated at period defined by the parameter 'measurement-interval'.

The test mode that has specific number of the test packets configured for the test session in the 'number-of-packets' parameter is referred as periodic mode. The test session may be repeated by the Session-Sender with the same parameters. The 'repeat' parameter defines number of tests and the 'repeat-interval' - the interval between the consecuitive tests. The performance metrics are calculated after each test session when the interval defined by the 'session-timeout' expires.

2.1.2. Session-Reflector

The twamp-light-session-reflector container holds items that are related to the configuration of the TWAMP-Light Session-Reflector logical entity.

The twamp-light-session-refl-state container holds Session-Reflector state data for the particular TWAMP-Light test session.

3. Data Model

Creating TWAMP-Light data model presents number of challenges and among them is identification of a test-session at Session-Reflector. A Session-Reflector MAY require only as little as its IP and UDP port number in received TWAMP-Test packet to spawn new test session. More so, to test processing of Class-of-Service along the same route in Equal Cost Multi-Path environment Session-Sender may run TWAMP test sessions concurrently using the same source IP address, source UDP port number, destination IP address, and destination UDP port number. Thus the only parameter that can be used to differentiate these test sessions would be DSCP value. The DSCP field may get re-marked along the path and without use of [RFC7750] that will go undetected, but by using five-tuple instead of four-tuple as a key we can ensure that TWAMP test packets that are considered as different test sessions follow the same path even in ECMP environments.

3.1. Tree Diagram

```
module: ietf-twamp-light
    +--rw twamp-light
    |  +--rw twamp-light-session-sender {session-sender-light}?
    |  |  +--rw sender-light-enable?   enable
    |  |  +--rw test-session* [session-id]
    |  |     +--rw session-id                      uint32
    |  |     +--rw test-session-enable?            enable
    |  |     +--rw number-of-packets?              union
    |  |     +--rw packet-padding-size?            uint32
    |  |     +--rw interval?                       uint32
    |  |     +--rw session-timeout?                uint32
    |  |     +--rw measurement-interval?           uint32
    |  |     +--rw repeat?                         union
    |  |     +--rw repeat-interval?                uint32
    |  |     +--rw dscp-value?                     inet:dscp
    |  |     +--rw test-session-reflector-mode?    session-reflector-mode
    |  |     +--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 authentication-params! {twamp-light-authentication}?
    |  |     |  +--rw key-chain?   kc:key-chain-ref
    |  |     +--rw first-percentile?               percentile
    |  |     +--rw second-percentile?              percentile
    |  |     +--rw third-percentile?               percentile
    |  +--rw twamp-light-session-reflector {session-reflector-light}?
    |     +--rw reflector-light-enable?       enable
    |     +--rw ref-wait?                     uint32
    |     +--rw reflector-light-mode-state?   session-reflector-mode
    |     +--rw test-session* [session-id]
    |        +--rw session-id               uint32
    |        +--rw dscp-handling-mode?      session-dscp-mode
    |        +--rw dscp-value?              inet:dscp
    |        +--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 authentication-params! {twamp-light-authentication}?
    |           +--rw key-chain?   kc:key-chain-ref
    +--ro twamp-light-state
       +--ro twamp-light-session-sender-state {session-sender-light}?
       |  +--ro test-session-state* [session-id]
       |     +--ro session-id              uint32
       |     +--ro sender-session-state?   enumeration
       |     +--ro current-stats
```

```
       |     |  +--ro start-time                yang:date-and-time
       |     |  +--ro packet-padding-size?      uint32
       |     |  +--ro interval?                 uint32
       |     |  +--ro duplicate-packets?        uint32
       |     |  +--ro reordered-packets?        uint32
       |     |  +--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 dscp?                     inet:dscp
       |     |  +--ro sent-packets?             uint32
       |     |  +--ro rcv-packets?              uint32
       |     |  +--ro sent-packets-error?       uint32
       |     |  +--ro rcv-packets-error?        uint32
       |     |  +--ro last-sent-seq?            uint32
       |     |  +--ro last-rcv-seq?             uint32
       |     |  +--ro two-way-delay
       |     |  |  +--ro delay
       |     |  |  |  +--ro min?   yang:gauge32
       |     |  |  |  +--ro max?   yang:gauge32
       |     |  |  |  +--ro avg?   yang:gauge32
       |     |  |  +--ro delay-variation
       |     |  |     +--ro min?   uint32
       |     |  |     +--ro max?   uint32
       |     |  |     +--ro avg?   uint32
       |     |  +--ro one-way-delay-far-end
       |     |  |  +--ro delay
       |     |  |  |  +--ro min?   yang:gauge32
       |     |  |  |  +--ro max?   yang:gauge32
       |     |  |  |  +--ro avg?   yang:gauge32
       |     |  |  +--ro delay-variation
       |     |  |     +--ro min?   uint32
       |     |  |     +--ro max?   uint32
       |     |  |     +--ro avg?   uint32
       |     |  +--ro one-way-delay-near-end
       |     |  |  +--ro delay
       |     |  |  |  +--ro min?   yang:gauge32
       |     |  |  |  +--ro max?   yang:gauge32
       |     |  |  |  +--ro avg?   yang:gauge32
       |     |  |  +--ro delay-variation
       |     |  |     +--ro min?   uint32
       |     |  |     +--ro max?   uint32
       |     |  |     +--ro avg?   uint32
       |     |  +--ro low-percentile
       |     |  |  +--ro delay-percentile
       |     |  |  |  +--ro rtt-delay?        percentile
       |     |  |  |  +--ro near-end-delay?   percentile
       |     |  |  |  +--ro far-end-delay?    percentile
```

```
|     |  |  +--ro jitter-percentile
|     |  |     +--ro rtt-jitter?        percentile
|     |  |     +--ro near-end-jitter?   percentile
|     |  |     +--ro far-end-jitter?    percentile
|     |  +--ro mid-percentile
|     |  |  +--ro delay-percentile
|     |  |  |  +--ro rtt-delay?        percentile
|     |  |  |  +--ro near-end-delay?   percentile
|     |  |  |  +--ro far-end-delay?    percentile
|     |  |  +--ro jitter-percentile
|     |  |     +--ro rtt-jitter?        percentile
|     |  |     +--ro near-end-jitter?   percentile
|     |  |     +--ro far-end-jitter?    percentile
|     |  +--ro high-percentile
|     |  |  +--ro delay-percentile
|     |  |  |  +--ro rtt-delay?        percentile
|     |  |  |  +--ro near-end-delay?   percentile
|     |  |  |  +--ro far-end-delay?    percentile
|     |  |  +--ro jitter-percentile
|     |  |     +--ro rtt-jitter?        percentile
|     |  |     +--ro near-end-jitter?   percentile
|     |  |     +--ro far-end-jitter?    percentile
|     |  +--ro two-way-loss
|     |  |  +--ro loss-count?         int32
|     |  |  +--ro loss-ratio?         percentage
|     |  |  +--ro loss-burst-max?     int32
|     |  |  +--ro loss-burst-min?     int32
|     |  |  +--ro loss-burst-count?   int32
|     |  +--ro one-way-loss-far-end
|     |  |  +--ro loss-count?         int32
|     |  |  +--ro loss-ratio?         percentage
|     |  |  +--ro loss-burst-max?     int32
|     |  |  +--ro loss-burst-min?     int32
|     |  |  +--ro loss-burst-count?   int32
|     |  +--ro one-way-loss-near-end
|     |     +--ro loss-count?         int32
|     |     +--ro loss-ratio?         percentage
|     |     +--ro loss-burst-max?     int32
|     |     +--ro loss-burst-min?     int32
|     |     +--ro loss-burst-count?   int32
|     +--ro history-stats* [id]
|        +--ro id                        uint32
|        +--ro end-time                  yang:date-and-time
|        +--ro number-of-packets?        uint32
|        +--ro packet-padding-size?      uint32
|        +--ro interval?                 uint32
|        +--ro duplicate-packets?        uint32
|        +--ro reordered-packets?        uint32
```

```
|        +--ro loss-packets?             uint32
|        +--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 dscp?                     inet:dscp
|        +--ro sent-packets?             uint32
|        +--ro rcv-packets?              uint32
|        +--ro sent-packets-error?       uint32
|        +--ro rcv-packets-error?        uint32
|        +--ro last-sent-seq?            uint32
|        +--ro last-rcv-seq?             uint32
|        +--ro two-way-delay
|        |  +--ro delay
|        |  |  +--ro min?   yang:gauge32
|        |  |  +--ro max?   yang:gauge32
|        |  |  +--ro avg?   yang:gauge32
|        |  +--ro delay-variation
|        |     +--ro min?   uint32
|        |     +--ro max?   uint32
|        |     +--ro avg?   uint32
|        +--ro one-way-delay-far-end
|        |  +--ro delay
|        |  |  +--ro min?   yang:gauge32
|        |  |  +--ro max?   yang:gauge32
|        |  |  +--ro avg?   yang:gauge32
|        |  +--ro delay-variation
|        |     +--ro min?   uint32
|        |     +--ro max?   uint32
|        |     +--ro avg?   uint32
|        +--ro one-way-delay-near-end
|           +--ro delay
|           |  +--ro min?   yang:gauge32
|           |  +--ro max?   yang:gauge32
|           |  +--ro avg?   yang:gauge32
|           +--ro delay-variation
|              +--ro min?   uint32
|              +--ro max?   uint32
|              +--ro avg?   uint32
|
+--ro twamp-light-session-refl-state {session-reflector-light}?
   +--ro reflector-light-admin-status    boolean
   +--ro test-session-state* [session-id]
      +--ro session-id            uint32
      +--ro sent-packets?         uint32
      +--ro rcv-packets?          uint32
      +--ro sent-packets-error?   uint32
      +--ro rcv-packets-error?    uint32
      +--ro last-sent-seq?        uint32
```

```
             +--ro last-rcv-seq?         uint32
             +--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

  rpcs:
    +---x twamp-sender-start
    |  +---w input
    |     +---w session-id    uint32
    +---x twamp-sender-stop
       +---w input
          +---w session-id    uint32
```

3.2. YANG Module

<CODE BEGINS> file "ietf-twamp-light@2017-06-13.yang"

```
module ietf-twamp-light {
   namespace "urn:ietf:params:xml:ns:yang:ietf-twamp-light";
    //namespace need to be assigned by IANA
   prefix "ietf-twamp-light";

   import ietf-inet-types {
     prefix inet;
   }
   import ietf-yang-types {
     prefix yang;
   }
   import ietf-key-chain {
     prefix kc;
   }

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

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

   description "TWAMP Light Data Model";

   revision "2017-06-13" {
     description
     "08 version. Appendix I RFC 5357 is covered.";
     reference "RFC 5357";
   }
```

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

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

   feature twamp-light-authentication {
     description
     "TWAMP Light authentication supported";
   }

   typedef enable {
      type boolean;
      description "enable";
   }

   typedef session-reflector-mode {
        type enumeration {
                enum stateful {
                        description
                        "When the Session-Reflector is stateful,
                        i.e. is aware of TWAMP-Test session state.";
                }
                enum stateless {
                        description
                        "When the Session-Reflector is stateless,
                        i.e. is not aware of the state of
                        TWAMP-Test session.";
                }
        }
        description "State of the Session-Reflector";
   }

   typedef session-dscp-mode {
        type enumeration {
                enum copy-received-value {
                        description
                        "Use DSCP value copied from received
                        TWAMP test packet of the test session.";
                }
                enum use-configured-value {
                        description
```

```
                        "Use DSCP value configured for this
                        test session on the Session-Reflector.";
                }
        }
        description
        "DSCP handling mode by Session-Reflector.";
   }

   typedef percentage {
        type decimal64 {
                fraction-digits 5;
        }
        description "Percentage";
   }

   typedef percentile {
        type decimal64 {
                fraction-digits 2;
        }
        description
        "Percentile is a measure used in statistics
        indicating the value below which a given
        percentage of observations in a group of
        observations fall.";
   }

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

```
     leaf last-rcv-seq {
       type uint32;
       description "Last received sequence number";
     }
   }

grouping twamp-session-percentile {
   description "Percentile grouping";
   leaf first-percentile {
      type percentile;
      default 95.00;
      description
      "First percentile to report";
   }
   leaf second-percentile {
      type percentile;
      default 99.00;
      description
      "Second percentile to report";
   }
   leaf third-percentile {
      type percentile;
      default 99.90;
      description
      "Third percentile to report";
   }
}

   grouping delay-statistics {
     description "Delay statistics grouping";
     container delay {
       description "Packets transmitted delay";
                 leaf min {
                   type yang:gauge32;
                   units microseconds;
                   description
                   "Min of Packets transmitted delay";
                 }
                 leaf max {
                    type yang:gauge32;
                    units microseconds;
                    description
                    "Max of Packets transmitted delay";
                 }
                 leaf avg {
                    type yang:gauge32;
                    units microseconds;
                    description
```

```
                   "Avg of Packets transmitted delay";
                }
     }

          container delay-variation {
                  description
                  "Packets transmitted delay variation";
                  leaf min {
                    type uint32;
                    units microseconds;
                    description
                    "Min of Packets transmitted
                    delay variation";
                  }
                  leaf max {
                    type uint32;
                    units microseconds;
                    description
                    "Max of Packets transmitted
                    delay variation";
                  }
                  leaf avg {
                    type uint32;
                    units microseconds;
                    description
                    "Avg of Packets transmitted
                    delay variation";
                  }
         }
   }
     grouping time-percentile-report {
        description "Delay percentile report grouping";
        container delay-percentile {
                description
                "Report round-trip, near- and far-end delay";
                leaf rtt-delay {
                        type percentile;
                        description
                        "Percentile of round-trip delay";
                }
                leaf near-end-delay {
                        type percentile;
                        description
                        "Percentile of near-end delay";
                }
                leaf far-end-delay {
                        type percentile;
                        description
```

```
                        "Percentile of far-end delay";
                }
        }
        container jitter-percentile {
                description
                "Report round-trip, near- and far-end jitter";
                leaf rtt-jitter {
                        type percentile;
                        description
                        "Percentile of round-trip jitter";
                }
                leaf near-end-jitter {
                        type percentile;
                        description
                        "Percentile of near-end jitter";
                }
                leaf far-end-jitter {
                        type percentile;
                        description
                        "Percentile of far-end jitter";
                }
        }
    }

    grouping packet-loss-statistics {
        description
        "Grouping for Packet Loss statistics";
        leaf loss-count {
                type int32;
                description
                "Number of lost packets
                during the test interval.";
        }
        leaf loss-ratio {
                type percentage;
                description
                "Ratio of packets lost to packets
                sent during the test interval.";
        }
        leaf loss-burst-max {
                type int32;
                description
                "Maximum number of consequtively
                lost packets during the test interval.";
        }
        leaf loss-burst-min {
                type int32;
                description
```

```
                "Minimum number of consequtively
                lost packets during the test interval.";
        }
                leaf loss-burst-count {
                type int32;
                description
                "Number of occasions with packet
                loss during the test interval.";
                }
    }

   grouping session-light-parameters {
     description
     "Parameters common among
     Session-Sender and Session-Reflector";
     leaf sender-ip {
       type inet:ip-address;
       mandatory true;
       description "Sender IP address";
     }
     leaf sender-udp-port {
       type inet:port-number {
         range "49152..65535";
       }
       mandatory true;
       description "Sender UDP port number";
     }
     leaf reflector-ip {
       type inet:ip-address;
       mandatory true;
       description "Reflector IP address";
     }
     leaf reflector-udp-port {
       type inet:port-number{
         range "49152..65535";
       }
       mandatory true;
       description "Reflector UDP port number";
     }
   }

   grouping session-light-auth-params {
      description
      "Grouping for TWAMP Light authentication parameters";
      container authentication-params {
         if-feature twamp-light-authentication;
         presence "Enables TWAMP Light authentication";
         description
```

```
         "Parameters for TWAMP Light authentication";
         leaf key-chain {
            type kc:key-chain-ref;
            description "Name of key-chain";
         }
      }
   }

   /*Configuration Data*/
   container twamp-light {
     description
     "Top level container for TWAMP-Light configuration";

    container twamp-light-session-sender {
       if-feature session-sender-light;
       description "TWAMP-Light Session-Sender container";

       leaf sender-light-enable {
         type enable;
         default "true";
         description
         "Whether this network element is enabled to
         act as TWAMP-Light Sender";
       }

       list test-session {
         key "session-id";
               unique "sender-ip sender-udp-port reflector-ip"
         +" reflector-udp-port dscp-value";
         description
         "This structure is a container of test session
         managed objects";

         leaf session-id {
           type uint32;
           description "Session ID";
         }

         leaf test-session-enable {
           type enable;
           default "true";
           description
           "Whether this TWAMP Test session is enabled";
         }

         leaf number-of-packets {
           type union {
                type uint32 {
```

```
                        range 1..4294967294 {
                                description
                                "The overall number of UDP test packet
                                to be transmitted by the sender for this
                                test session";
                        }
                }
                type enumeration {
                        enum forever {
                                description
                                "Indicates that the test session SHALL
                                be run *forever*.";
                        }
                }
          }
          default 10;
          description
          "This value determines if the TWAMP-Test session is
             bound by number of test packets or not.";
        }

        leaf packet-padding-size {
          type uint32;
          default 27;
          description
          "Size of the Packet Padding. Suggested to run
          Path MTU Discovery to avoid packet fragmentation in
          IPv4 and packet blackholing in IPv6";
        }

        leaf interval  {
          type uint32;
          units microseconds;
          description
          "Time interval between transmission of two
          consecutive packets in the test session in
          microseconds";
        }

           leaf session-timeout {
             when "../number-of-packets != 'forever'" {
               description
               "Test session timeout only valid if the
               test mode is periodic.";
          }
          type uint32;
          units "seconds";
          default 900;
```

```
          description
          "The timeout value for the Session-Sender to
          collect outstanding reflected packets.";
        }

        leaf measurement-interval {
          when "../number-of-packets = 'forever'" {
               description
               "Valid only when the test to run forever,
               i.e. continuously.";
          }
          type uint32;
          units "seconds";
          default 60;
          description
          "Interval to calculate performance metric when
                the test mode is 'continuous'.";
           }

           leaf repeat {
             type union {
               type uint32 {
                       range 0..4294967294;
               }
               type enumeration {
                       enum forever {
                               description
                               "Indicates that the test session SHALL
                               be repeated *forever* using the
                               information in repeat-interval
                               parameter, and SHALL NOT decrement
                               the value.";
                       }
               }
             }
             default 0;
             description
             "This value determines if the TWAMP-Test session must
             be repeated. When a test session has completed, the
             repeat parameter is checked. The default value
             of 0 indicates that the session MUST NOT be repeated.
             If the repeat value is 1 through 4,294,967,294
             then the test session SHALL be repeated using the
             information in repeat-interval parameter.
             The implementation MUST decrement the value of repeat
             after determining a repeated session is expected.";
           }
```

```
            leaf repeat-interval {
                when "../repeat != '0'";
                type uint32;
                units seconds;
                default 0;
                description
                "This parameter determines the timing of repeated
                TWAMP-Test sessions when repeat is more than 0.";
            }

            leaf dscp-value {
              type inet:dscp;
              default 0;
              description
              "DSCP value to be set in the test packet.";
            }

            leaf test-session-reflector-mode {
                 type session-reflector-mode;
                 default "stateless";
                 description
                 "The mode of TWAMP-Reflector for the test session.";
            }

          uses session-light-parameters;
          uses session-light-auth-params;
          uses twamp-session-percentile;
        }
      }

      container twamp-light-session-reflector {
        if-feature session-reflector-light;
        description
        "TWAMP-Light Session-Reflector container";
          leaf reflector-light-enable {
            type enable;
            default "true";
            description
            "Whether this network element is enabled to
            act as TWAMP-Light Reflector";
          }

          leaf ref-wait {
          type uint32 {
            range 1..604800;
          }
          units seconds;
          default 900;
```

```
         description
         "REFWAIT(TWAMP test session timeout in seconds),
         the default value is 900";
       }

       leaf reflector-light-mode-state {
         type session-reflector-mode;
         default stateless;
         description
         "The state of the mode of the TWAMP-Light
         Session-Reflector";
        }

        list test-session {
          key "session-id";
                unique "sender-ip sender-udp-port reflector-ip"
          +" reflector-udp-port";
          description
          "This structure is a container of test session
          managed objects";

        leaf session-id {
          type uint32;
          description "Session ID";
        }

        leaf dscp-handling-mode {
          type session-dscp-mode;
          default copy-received-value;
          description
          "Session-Reflector handling of DSCP:
                 - use value copied from received TWAMP-Test packet;
                 - use value explicitly configured";
           }

           leaf dscp-value {
             when "../dscp-handling-mode = 'use-configured-value'";
             type inet:dscp;
             default 0;
             description
             "DSCP value to be set in the reflected packet
                if dscp-handling-mode is set to use-configured-value.";
           }

        uses session-light-parameters;
        uses session-light-auth-params;
       }
     }
```

```
   }

 /*Operational state data nodes*/
 container twamp-light-state{
   config "false";
   description
   "Top level container for TWAMP-Light state data";

   container twamp-light-session-sender-state {
     if-feature session-sender-light;
     description
     "Session-Sender container for state data";
     list test-session-state{
       key "session-id";
       description
       "This structure is a container of test session
       managed objects";

       leaf session-id {
         type uint32;
         description "Session ID";
       }

       leaf sender-session-state {
         type enumeration {
           enum active {
             description "Test session is active";
           }
           enum ready {
             description "Test session is idle";
           }
         }
         description
         "State of the particular TWAMP-Light test
         session at the sender";
       }

       container current-stats {
         description
         "This container contains the results for the current
          Measurement Interval in a Measurement session ";
          leaf start-time {
            type yang:date-and-time;
            mandatory true;
            description
            "The time that the current Measurement Interval started";
          }
```

```
          leaf packet-padding-size {
            type uint32;
            default 27;
            description
            "Size of the Packet Padding. Suggested to run
            Path MTU Discovery to avoid packet fragmentation
            in IPv4 and packet backholing in IPv6";
          }

          leaf interval  {
            type uint32;
            units microseconds;
           description
           "Time interval between transmission of two
           consecutive packets in the test session";
          }

          leaf duplicate-packets  {
            type uint32;
           description "Duplicate packets";
          }
          leaf reordered-packets  {
            type uint32;
           description "Reordered packets";
          }

          uses session-light-parameters;
          leaf dscp {
            type inet:dscp;
            description
            "The DSCP value that was placed in the header of
            TWAMP UDP test packets by the Session-Sender.";
          }
          uses maintenance-statistics;

          container two-way-delay {
            description
            "two way delay result of the test session";
            uses delay-statistics;
          }

          container one-way-delay-far-end {
            description
            "one way delay far-end of the test session";
            uses delay-statistics;
          }

          container one-way-delay-near-end {
```

```
            description
            "one way delay near-end of the test session";
            uses delay-statistics;
          }

          container low-percentile {
                when "/twamp-light/twamp-light-session-sender/"
                +"test-session[session-id]/"
                +"first-percentile != '0.00'" {
                        description
                        "Only valid if the
                        the first-percentile is not NULL";
                }
                description
                "Low percentile report";
                uses time-percentile-report;
          }

          container mid-percentile {
                when "/twamp-light/twamp-light-session-sender/"
                +"test-session[session-id]/"
                +"second-percentile != '0.00'" {
                        description
                        "Only valid if the
                        the first-percentile is not NULL";
                }
                description
                "Mid percentile report";
                uses time-percentile-report;
          }

          container high-percentile {
                when "/twamp-light/twamp-light-session-sender/"
                +"test-session[session-id]/"
                +"third-percentile != '0.00'" {
                        description
                        "Only valid if the
                        the first-percentile is not NULL";
                }
                description
                "High percentile report";
                uses time-percentile-report;
          }

          container two-way-loss {
            description
            "two way loss count and ratio result of
            the test session";
```

```
            uses packet-loss-statistics;
          }
          container one-way-loss-far-end {
            when "/twamp-light/twamp-light-session-sender/"
            +"test-session[session-id]/"
            +"test-session-reflector-mode = 'stateful'" {
                description
                "One-way statistic is only valid if the
                session-reflector is in stateful mode.";
            }
            description
            "one way loss count and ratio far-end of
            the test session";
            uses packet-loss-statistics;
          }
          container one-way-loss-near-end {
            when "/twamp-light/twamp-light-session-sender/"
            +"test-session[session-id]/"
            +"test-session-reflector-mode = 'stateful'" {
                description
                "One-way statistic is only valid if the
                session-reflector is in stateful mode.";
            }
            description
            "one way loss count and ratio near-end of
            the test session";
            uses packet-loss-statistics;
          }
        }

        list history-stats {
          key id;
          description
          "This container contains the results for the history
           Measurement Interval in a Measurement session ";
           leaf id {
             type uint32;
             description
             "The identifier for the Measurement Interval
             within this session";
           }
           leaf end-time {
             type yang:date-and-time;
             mandatory true;
             description
             "The time that the Measurement Interval ended";
           }
           leaf number-of-packets {
```

```
            type uint32;
            description
            "The overall number of UDP test packets to be
            transmitted by the sender for this test session";
          }

          leaf packet-padding-size {
            type uint32;
            default 27;
            description
            "Size of the Packet Padding. Suggested to run
            Path MTU Discovery to avoid packet fragmentation
            in IPv4 and packet blackholing in IPv6";
          }

          leaf interval  {
            type uint32;
            units microseconds;
           description
           "Time interval between transmission of two
           consecutive packets in the test session";
          }
          leaf duplicate-packets  {
            type uint32;
           description "Duplicate packets";
          }
          leaf reordered-packets  {
            type uint32;
           description "Reordered packets";
          }
          leaf loss-packets  {
            type uint32;
           description "Loss packets";
          }

          uses session-light-parameters;
          leaf dscp {
            type inet:dscp;
            description
            "The DSCP value that was placed in the header of
            TWAMP UDP test packets by the Session-Sender.";
          }
          uses maintenance-statistics;

          container two-way-delay{
            description
            "two way delay result of the test session";
            uses delay-statistics;
```

```
          }
          container one-way-delay-far-end{
            description
            "one way delay far end of the test session";
            uses delay-statistics;
          }
          container one-way-delay-near-end{
            description
            "one way delay near end of the test session";
            uses delay-statistics;
          }
        }
      }
    }

     container twamp-light-session-refl-state {
       if-feature session-reflector-light;
       description
       "TWAMP-Light Session-Reflector container for
       state data";
       leaf reflector-light-admin-status {
         type boolean;
         mandatory "true";
         description
         "Whether this network element is enabled to
         act as TWAMP-Light Reflector";
       }

      list test-session-state {
        key "session-id";
        description
        "This structure is a container of test session
        managed objects";

        leaf session-id {
          type uint32;
          description "Session ID";
        }

        uses maintenance-statistics;
        uses session-light-parameters;
      }
    }
  }

  rpc twamp-sender-start {
    description
          "start the configured sender session";
```

```
   input {
         leaf session-id {
           type uint32;
           mandatory true;
           description
                 "The session to be started";
         }
   }
 }

 rpc twamp-sender-stop {
   description
         "stop the configured sender session";
   input {
         leaf session-id {
           type uint32;
           mandatory true;
           description
                 "The session to be stopped";
         }
   }
 }
}
```

<CODE ENDS>

4. IANA Considerations

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

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

Registrant Contact: The IPPM WG of the IETF.

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

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

name: ietf-twamp-light

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

prefix: twamp

reference: RFC XXXX

5. Security Considerations

The configuration, state, action data defined in this document may be accessed via the NETCONF protocol [RFC6241]. SSH [RFC6242] is mandatory secure transport that is the lowest NETCONF layer. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

But, in general, this TWAMP Light YANG module does not change any underlying security issues that already may exist in [I-D.elteto-ippm-twamp-mib].

6. References

6.1. Normative References

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

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

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

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

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

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

6.2. Informative References

[I-D.elteto-ippm-twamp-mib]
Elteto, T. and G. Mirsky, "Two-Way Active Measurement Protocol (TWAMP) Management Information Base (MIB)", draft-elteto-ippm-twamp-mib-01 (work in progress), January 2014.

[I-D.ietf-ippm-twamp-yang]
Civil, R., Morton, A., Rahman, R., Jethanandani, M., and K. Pentikousis, "Two-Way Active Measurement Protocol (TWAMP) Data Model", draft-ietf-ippm-twamp-yang-03 (work in progress), February 2017.

Appendix A. Acknowledgements

TBD

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Xiao Min
ZTE Corp.

Email: xiao.min2@zte.com.cn

Adrian Pan
Ericsson

Email: adrian.pan@ericsson.com

Wei S Luo
Ericsson

Email: wei.s.luo@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2017

T. Mizrahi
Marvell
G. Fioccola
Telecom Italia
M. Chen
L. Zheng
Huawei Technologies
G. Mirsky
March 5, 2017

Passive Performance Monitoring using a Multiplexed Marking Field
draft-mizrahi-ippm-multiplexed-alternate-marking-01

Abstract

This memo introduces a marking method that uses a single marking bit, or two marking values, and allows accurate loss and delay measurement.

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 6, 2017.

Copyright Notice

Table of Contents

1. Introduction

Alternate marking, defined in [I-D.ietf-ippm-alt-mark], is a method for measuring packet loss, packet delay, and packet delay variation. Typical delay measurement protocols require the two measurement points (MPs) to exchange timestamped test packets. In contrast, the alternate marking method does not require control packets to be exchanged. Instead, every data packet carries a color indicator, which divides the traffic into consecutive blocks of packets.

The color indicator may either be a single-bit binary indication, or a two reserved values of a larger field, such as an IPv6 Flow Label or an MPLS Label. Throughout the rest of the document it is assumed that the color indication is a single-bit field, unless specified otherwise. The color value is toggled periodically, as illustrated in Figure 1.

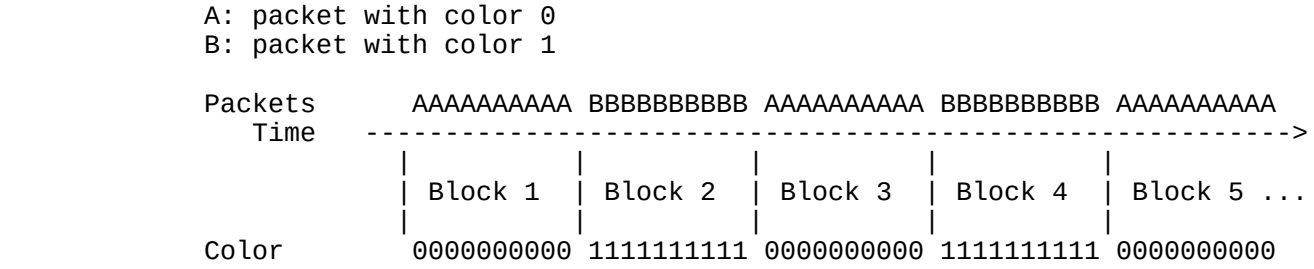


Figure 1: Alternate marking: packets are monitored on a per-color basis.

Alternate marking is used between two MPs, the initiating MP, and the monitoring MP. The initiating MP incorporates the marking field into en-route packets, allowing the monitoring MP to use the marking field in order to bind each packet to the corresponding block.

Each of the MPs maintains two counters, one per color. At the end of each block the counter values can be collected by a central management system, and analyzed; the packet loss can be computed by comparing the counter values of the two MPs.

When using alternate marking delay measurement can be performed in one of three ways (as per [I-D.ietf-ippm-alt-mark]):

- Single marking: the first packet of each block is used by both MPs as a reference for delay measurement. The timestamp of this packet is measured by the two measurement points, and can be collected by the mangement system from each of the measurement points, which can compute the path delay by comparing the two timestamps. The drawback of this approach is that it is not accurate when packets arrive out-of-order, as the two measurement may have a different view of which packet was the first in the block.

- Average delay: each of the MPs computes the average packet timestamp of each block. The management system can then compute the delay by comparing the average times of the two MPs. The drawback of this approach is that it may be computationally heavy, or difficult to implement at the data plane.

- Double marking: each packet uses two marking bits. One bit is used as a color indicator, and one is used as a timestamping indicator. This method resolves the drawbacks raised for the two previous methods, at the expense of an extra bit in the packet header.

The double marking method allows for accurate measurement without incurring expensive computational load. However, in some cases allocating two bits for passive measurement is not possible. For example, if alternate marking is implemented over IPv4, allocating 2 marking bits in the IPv4 header is challenging, as every bit in the 20-octet header is costly; one of the possible approaches discussed in [I-D.ietf-ippm-alt-mark] is reserve one or two bits from the DSCP field for remarking. In this case every marking bit comes at the expense of reducing the DSCP range by a factor of two.

This memo extends the marking method of [I-D.ietf-ippm-alt-mark]. The method introduced in this document uses a single marking bit in the packet header, while providing the advantages of the double marking method. In a nutshell, the color indicator and the timestamp indicator are multiplexed into a single bit. There is an underlying assumption that the two MPs that take part in the measurement are time-synchronized.

2. Terminology

2.1. 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].

2.2. Abbreviations

MP Measurement Point

MPLS Multiprotocol Label Switching

DSCP Differentiated Services Code Point

LSP Label Switched Path

SFL Synonymous Flow Label [I-D.bryant-mpls-sfl-framework]

3. Alternate Marking using a Multiplexed Marking Bit

3.1. Overview

This section introduces a method that uses a single marking bit that serves two purposes: a color indicator, and a timestamp indicator. The double marking method that was discussed in the previous section uses two 1-bit values: a color indicator C, and a timestamp indicator T. The multiplexed marking bit, denoted by M, is an exclusive or between these two values: M = C XOR T.

An example of the use of the multiplexed marking bit is depicted in Figure 2. The example considers two routers, R1 and R2, that use the multiplexed bit method to measure traffic from R1 to R2. In each block R1 designates one of the packets for delay measurement. In each of these designated packets the value of the multiplexed bit is reversed compared to the other packets in the same block, allowing R2 to distinguish the designated packets from the other packets.

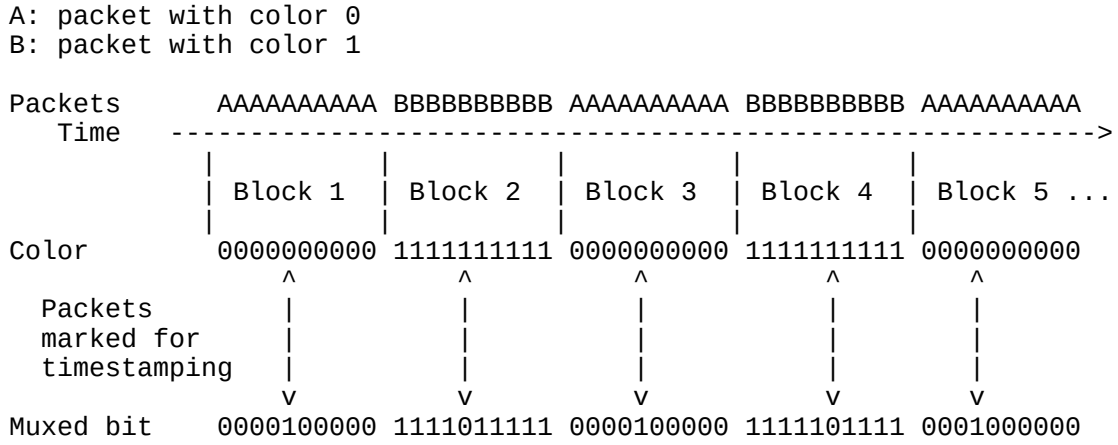


Figure 2: Alternate marking with multiplexed bit.

3.2. Timing and Synchronization Aspects

It is assumed that all MPs are synchronized to a common reference time with an accuracy of +/- A/2. Thus, the difference between the clock values of any two MPs is bounded by A. Clocks can be synchronized for example using NTP [RFC5905], PTP [IEEE1588], or by other means. The common reference time is used for dividing the time domain into equal-sized measurement periods, such that all packets forwarded during a measurement period have the same color, and consecutive periods have alternating colors.

The single marking bit incorporates two multiplexed values. From the monitoring MP's perspective, the two values are Time-Division Multiplexed (TDM), as depicted in Figure 3. It is assumed that the start time of every measurement period is known to both the initiating MP and the monitoring MP. If the measurement period is L, then during the first and the last L/4 time units of each block the marking bit is interpreted by the monitoring MP as a color indicator. During the middle part of the block, the marking bit is interpreted as a timestamp indicator; if the value of this bit is different than

the color value, the corresponding packet is used as a reference for delay measurement.

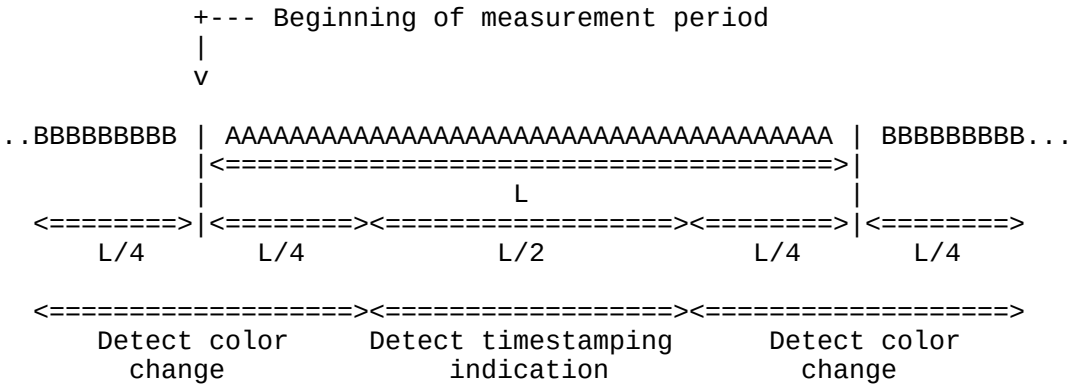


Figure 3: Multiplexed marking field interpretation at the receiving measurement point.

In order to prevent ambiguity in the receiver's interpretation of the marking field, the initiating MP is permitted to set the timestamp indication only during a specific interval, as depicted in Figure 4. Since the receiver is willing to receive the timestamp indication during the middle L/2 time units of the block, the sender refrains from sending the timestamp indication during a guardband interval of d time units at the beginning and end of the L/2-period.

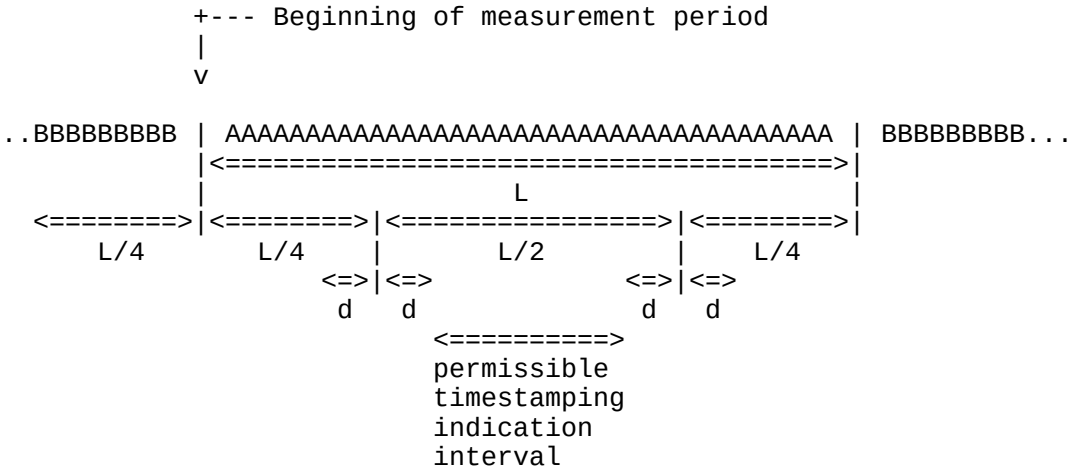


Figure 4: A time domain view.

The guardband d is given by d = A + D_max - D_min, where A is the clock accuracy, D_max is an upper bound on the network delay between the MPs, and D_min is a lower bound on the delay. It is straightforward from Figure 4 that d < L/4 must be satisfied. The latter implies a minimal requirement on the synchronization accuracy.

All MPs must be synchronized to the same reference time with an accuracy of +/- L/8. Depending on the system topology, in some systems the accuracy requirement will be even more stringent, subject to d < L/4. Note that the accuracy requirement of the conventional alternate marking method [I-D.ietf-ippm-alt-mark] is +/- L/2, while the multiplexed marking method requires an accuracy of +/- L/8.

Note that we assume that the middle L/2-period is designated as the timestamp indication period, allowing a sufficiently long guardband between the transitions. However, a system may be configured to use a longer timestamp indication period or a shorter one, if it is guaranteed that the synchronization accuracy meets the guardband requirements (i.e., the constraints on d).

4. Alternate Marking using Two Multiplexed Marking Values

As mentioned above, the color indicator is not necessarily a single bit, but may be implemented by using two well-known values in one of the header fields. For example, as defined in [I-D.bryant-mpls-rfc6374-sfl], two MPLS Label values can be used to indicate the two colors of a given LSP: the original Label value, and a Synonymous Flow Label (SFL) value.

The bit multiplexing approach of Section 3 is applicable not only to single-bit color indicators, but also to two-value indicators; instead of using a single bit that is toggled between ’0’ and ’1’, two values of the indicator field, U and W, can be used in the same manner, allowing both loss and delay measurement to be performed using only two reserved values. Thus, the multiplexing approach of Figure 2 can be illustrated more generally with two values, U and W, as depicted in Figure 5.

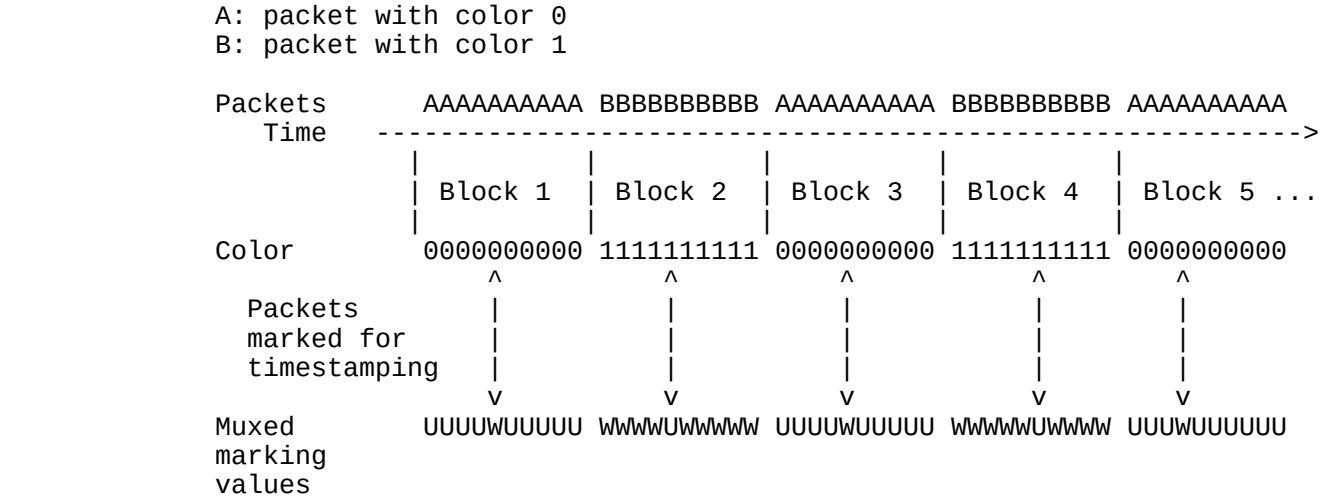


Figure 5: Alternate marking with two multiplexed marking values, U and W.

5. IANA Considerations

This memo includes no requests from IANA.

6. Security Considerations

The security considerations of the alternate marking method are discussed in [I-D.ietf-ippm-alt-mark]. Specifically, the method that is defined in this document requires slightly more stringent synchronization than the conventional marking method, potentially making the method more vulnerable to attacks on the time synchronization protocol. A detailed discussion about the threats against time protocols and how to mitigate them is presented in [RFC7384].

7. References

7.1. Normative References

[I-D.ietf-ippm-alt-mark]
Fioccola, G., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate Marking method for passive performance monitoring", draft-ietf-ippm-alt-mark-04 (work in progress), March 2017.

[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>.

7.2. Informative References

[I-D.bryant-mpls-rfc6374-sfl] Bryant, S., Chen, M., Li, Z., Swallow, G., Sivabalan, S., Mirsky, G., and G. Fioccola, "RFC6374 Synonymous Flow Labels", draft-bryant-mpls-rfc6374-sfl-03 (work in progress), October 2016.

[I-D.bryant-mpls-sfl-framework] Bryant, S., Chen, M., Li, Z., Swallow, G., Sivabalan, S., and G. Mirsky, "Synonymous Flow Label Framework", draft-bryant-mpls-sfl-framework-02 (work in progress), October 2016.

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

[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>.

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

Authors' Addresses

Tal Mizrahi
Marvell
6 Hamada st.
Yokneam
Israel

Email: talmi@marvell.com

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

Email: giuseppe.fioccola@telecomitalia.it

Mach(Guoyi) Chen
Huawei Technologies

Email: mach.chen@huawei.com

Lianshu Zheng
Huawei Technologies

Email: vero.zheng@huawei.com

Greg Mirsky
USA

Email: gregimirsky@gmail.com

IP Performance Working Group
Internet-Draft
Intended status: Informational
Expires: August 28, 2017

K. Nieminen
FICORA

February 27, 2017

Net Neutrality Measurements: Regulatory Use Case and Problem Statement
draft-nieminen-ippm-nn-measurements-00.txt

Abstract

This document describes a regulatory use case for net neutrality measurements based on the new European open internet regulation [1]. The purpose of this document is to give sufficient details for developing the actual net neutrality measurement metrics.

This document describes the problem statement. According to the Regulation European regulators has to supervise and enforce the net neutrality obligations. Especially the reliability of measurement results is important. However, monitoring net neutrality is a complex topic lacking standardized measurements.

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 28, 2017.

Copyright Notice

Table of Contents

1. Introduction

According to the European open internet regulation [1], providers of internet access services shall treat all traffic equally, when providing internet access services, without discrimination, restriction or interference, and irrespective of the sender and receiver, the content accessed or distributed, the applications or services used or provided, or the terminal equipment used.

The Regulation allows only few exceptions for this rule that are subject to strict interpretation and to proportionality requirements.

The Regulation imposes an obligation for European regulators to closely monitor and ensure compliance with the Regulation. Regulators must also promote the continued availability of non-discriminatory internet access services at levels of quality that reflect advances in technology.

The Regulation imposes also new transparency obligations for internet access service contract conditions. Regarding fixed networks, internet access service providers must publish among other things a clear and comprehensible explanation of the minimum, normally available, maximum and advertised download and upload speeds.

The Body of European Regulators for Electronic Communications (BEREC) has given guidelines on implementation of the Regulation [2]. The guidelines provide further information regarding the regulatory use cases.

This document strives to provide sufficient details about regulatory use cases for developing the actual net neutrality measurement metrics.

Although legal challenges can change the status of policy, the take-away for IPPM purposes is that many policy-makers are looking for measurement solutions to assist them in discovering discriminatory treatment of traffic flows. The exact definitions and requirements vary from one jurisdiction to another.

2. Problem statement

The regulators have a need to reliably assess violation of net neutrality with respect to the recently published BEREC guidelines on net neutrality and the underpinning EU legislation. In the broader sense, the regulators' need is to determine whether illegal traffic management practices is being applied to end-user traffic as per application, as well as monitor the evolution of the performance of the internet access service (IAS) over time.

It is envisaged that in order to carry out such assessment reliably, a reliable technical measurement of end-user Internet traffic behaviour needs to be conducted.

In 2015, Ofcom (communications regulator in the UK) commissioned a study to better understand the existing techniques that could be potentially used to detect traffic management. The study identified a number of techniques that have been developed and that are able to detect presence of particular kinds of differential traffic management. The study also found that a gap exists for effective detection of the presence of traffic management along the digital delivery chain, but that a potential standardized solution may still be possible. The study concluded that further work is required to develop a broader framework for traffic management detection solution.

When measurement tasks are run by an end-user, end-user environment specific factors like cross-traffic, measurement interface (fixed/wireless), firewalls, client operating system and hardware can influence the measurement result. These factors have to be detected and taken into account when assessing measurements performed by end-users. The topic is discussed further under Section 3.4.

According to BEREC guidelines speed should be calculated based on IP packet payload. This has to be taken into account when specifying the measurement metrics. It is noted that support for raw sockets is not universally available to standard users on most operating systems. For software based crowdsourcing approach it is essential that measurements can be performed using all common operating systems. Measurements should also support measurement client software installed by the end-user and as well as web browser based measurements.

The European Regulation requires internet service providers (ISPs) to specify new speed values for example minimum, maximum, and normally available speeds in fixed network. The measurement use case is to assess if these contractual speed values are met. The problem is to define measurements that can be run by end-users and is accurate enough to have legal value.

In addition to the mandatory requirements there are features that should be taken into account when planning the measurements to make them more usable and user friendly such as that the measurement does not block the internet access usage for whole day and does not generate excessive network load.

In principle, any solution should be equally applicable to both fixed and mobile Internet access services from narrow band to multi-gigabit connections. Certain variations may be accepted if they can be justified.

3. Regulatory use cases

Some regulatory use cases are already listed at a high level in RFC 7536 [3]. The purpose is to build on these use cases to further elaborate what is needed to fulfil EU regulatory requirements in view of the recent EU legislation and BEREC guidelines publications. This document targets the level of detail that is sufficient for developing actual measurement metrics and methodologies.

The goal of this document is to help to understand what metrics need to be defined and how they should be measured in order to produce repeatable results with high degree of accuracy. This document also gives a high level explanation of how these measurement tasks and results can be used for assessing net neutrality in different regulatory use cases.

The identified high-level measurement tasks are:

- Monitoring the performance of internet access services
- Detecting traffic management practices that impact the availability of individual applications
- Detecting traffic management practices that impact the quality of service for individual applications
- Detecting end-user dependent factors that may impact the measurement results

An end-user should be able to run a measurement process from an appropriate client. A regulator may provide additional complementary tests as part of a larger suite of testing. Both panel based and crowdsource solutions could be considered. It is possible to use both active and passive measurements to fulfil the regulatory requirements.

The solutions should be based on a minimum measurement time and data volume in order to ensure the validity of the measurements while taking care to avoid the possibility of harmful effect on end-user’s Internet consumption.

3.1. Monitoring the performance of internet access services

This use case is used to measure speed and other relevant internet access service (IAS) quality of service (QoS) parameters (e.g. delay, jitter and packet loss) for the IAS as a whole. It enables end-users to check their individual internet access speed and whether the IAS performance meets what has been specified in the contract. This has traditionally been the main motivation for end-users to use the tool provided by regulators.

Regulators may also run these measurements independently of any net neutrality assessment and use this information for multiple purposes such as increasing transparency in service provisioning (e.g. coverage maps) and monitoring the overall IAS quality, which may be aimed at:

- Ascertaining whether (or not) specialised services are provided at the expense of IAS, and/or

- Determining whether IAS performance is evolving in tandem with advances in technology.

The European open internet regulation [1] states that an end-user may use a monitoring mechanism certified by the regulator to check that the performance meets what has been specified in the contract. This measurement information can be used for triggering the remedies available to the consumer in accordance with national law.

BEREC guidelines [2] defines further that the certified monitoring mechanism should mitigate, to the extent possible, confounding factors which are internal to the user environment. Examples of these factors include existing cross-traffic and the usage of wireless/wireline interfaces.

According to BEREC guidelines speed should be calculated based on IP packet payload. Measurements should also be performed beyond the internet service provider (ISP) leg.

According to the Regulation ISPs must specify the minimum, normally available, maximum and advertised download and upload speed in their fixed network contracts. For mobile network subscriptions ISPs must specify estimated maximum and advertised download and upload speeds.

According to the recitals of the Regulation the normally available speed is understood to be the speed that an end-user could expect to receive most of the time when accessing the service. BEREC has given further guidance that the speed should be available during the specified daily period. For example a regulator may set a requirement that the normally available speed should be available during off-peak hours and 90% of time over peak hours, or 95% over the whole day.

Other factors that require special attention are how the minimum and maximum speed should be measured. According to BEREC guidelines the

- maximum speed is the speed that an end-user could expect to receive at least some of the time (e.g. at least once a day).
- minimum speed is the lowest speed that the ISP undertakes to deliver to the end-user. In principle, the actual speed should not be lower than the minimum speed, except in cases of interruption of the IAS.

Number and distribution of measurement tasks should be defined so that the adequate confidence level such as 95% is achieved.

3.2. Detecting traffic management practices that impact the availability of individual applications

The goal of this use case is to detect traffic management practices that affect the connectivity and availability of content, applications and services. Examples of this kind of practices may include blocking communication ports, VoIP and P2P file sharing applications in addition to other web content like streaming services, network based parental control and ad-blocking.

Internet service providers may use several different traffic management practices that block the connectivity to content, applications and services. Examples of these traffic management practices include: - Blocked communication ports

- IP addresses blocking
- DNS manipulation and HTTP proxy blocking
- Content or application based blocking with deep packet inspection

The challenge is to define specific measurement tasks that allow regulators to detect any blocked applications. A solution should minimise the probability of false positives. In principle, the solution is to comprise of measurement metric(s) and respective measurement methodology(s); as well as quantification of the probability of false positives.

3.3. Detecting traffic management practices that impact the quality of service for individual applications

The goal of this use case is to detect possible unequal treatment of traffic namely prioritisation and/or throttling of applications.

These traffic management practices may be detected by measuring the QoS experienced by the application and comparing the results with the QoS measurement results for the same IAS subscriptions and with the similar application specific QoS measurement results from other users and ISPs. Other techniques may also be possible.

A solution is required that correctly identifies whether prioritisation and/or throttling of applications is taking place, with minimum probability of false positives. In principle, the solution is to comprise of measurement metric(s) and respective measurement methodology(s); as well as quantification of the probability of false positives.

For this case, in particular, regulator may need to conduct additional complementary measurement tasks as part of a larger suite of testing, in order to eliminate any false positives.

3.4. Detecting end-user dependent factors that may impact the measurement results

Especially when measurements are run by an end-user in a crowdsourcing measurement setup, the local environment specific factors like cross-traffic, interface type (fixed/wireless), firewalls, processor load, client operating system and hardware can influence the measurement result.

It is preferred that the measurement client should capture this additional data of the end user local environment. This environment data can then be used in assessing the validity of the measurement results with the aim of improving overall accuracy and minimising false positives.

In principle, the solution is to comprise of measurement metric(s) and respective measurement methodology(s), and how this environment data can be used to ascertain measurement reliability in each use case.

4. Security Considerations

This document defines a use case and problem statement for net neutrality measurements. Security considerations for specific measurements will be discussed in solution documents.

5. IANA Considerations

This document includes no requests to the IANA.

6. References

6.1. Informative References

[1] Regulation (EU) 2015/2120 of the European Parliament and of the Council of 25 November 2015 laying down measures concerning open internet access and amending Directive 2002/22/EC on universal service and users' rights relating to electronic communications networks and services and Regulation (EU) No 531/2012 on roaming on public mobile communications networks within the Union, November 2015, <http://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:32015R2120>.

[2] BEREC Guidelines on the Implementation by National Regulators of European Net Neutrality Rules, August 2016, <http://berec.europa.eu/eng/document_register/subject_matter/berec/regulatory_best_practices/guidelines/6160-berec-guidelines-on-the-implementation-by-national-regulators-of-european-net-neutrality-rules>.

[3] Linsner, M., Eardley, P., Burbridge, T. and Sorensen, F., "Large-Scale Broadband Measurement Use Cases", RFC 7536, May 2015, <http://www.rfc-editor.org/info/rfc7536>.

7. Acknowledgements

The author wish to thank Ahmed Aldabbagh, Mick Fox, Jose Hernan, Frode Sorensen and Volker Sypli for their invaluable comments and contributions.

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

Author's Address

Klaus Nieminen
FICORA
Itamerenkatu 3 A, P.O Box 313
Finland

Email: klaus.nieminen@ficora.fi