K. Hedayat
Brix Networks
R. Krzanowski
Verizon
K. Yum
Juniper Networks
A. Morton
AT&T Labs
J. Babiarz
Nortel Networks

Internet Draft
Expires: September 2007
March 2007

A Two-way Active Measurement Protocol (TWAMP)
draft-ietf-ippm-twamp-03

Status of this Memo

Copyright Notice

Abstract

The IPPM One-way Active Measurement Protocol [RFC4656] (OWAMP) provides a common protocol for measuring one-way metrics between network devices. OWAMP can be used bi-directionally to measure one-way metrics in both directions between two network elements. However, it does not accommodate round-trip or two-way measurements. This memo specifies a Two-way Active Measurement Protocol (TWAMP), based on the OWAMP, that adds two-way or round-trip measurement capabilities. The TWAMP measurement architecture is usually comprised of two hosts with specific roles, and this allows for some protocol simplifications, making it an attractive alternative in some circumstances.

Table of Contents

1. Introduction

The IETF IP Performance Metrics (IPPM) working group has completed a draft standard for the round-trip delay [RFC2681] metric. IPPM has also completed a protocol for the control and collection of one-way measurements, the One-way Active Measurement Protocol (OWAMP) [RFC4656]. However, OWAMP does not accommodate round-trip or two-way measurements.

Two-way measurements are common in IP networks, primarily because time accuracy is less demanding for round-trip delay, and measurement support at the remote end may be limited to a simple echo function. This memo specifies the Two-way Active Measurement Protocol, or TWAMP. TWAMP uses the methodology and architecture of OWAMP [RFC4656] to define an open protocol for measurement of two-way or round-trip metrics (henceforth in this document the term two-way also signifies round-trip). The TWAMP measurement architecture is usually comprised of only two hosts with specific roles, and this allows for some protocol simplifications, making it an attractive alternative in some circumstances.

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

1.1 Relationship of Test and Control Protocols

Similar to OWAMP [RFC4656], TWAMP consists of two inter-related protocols: TWAMP-Control and TWAMP-Test. The relationship of these protocols is as defined in section 1.1 of OWAMP [RFC4656].

1.2 Logical Model

The role and definition of the logical entities are as defined in section 1.2 of OWAMP [RFC4656] with the following exceptions:

- The Session-Receiver is called the Session-Reflector in the TWAMP architecture. The Session-Reflector has the capability to create and send a measurement packet when it receives a measurement packet. Unlike the Session-Receiver, the Session-Reflector does not collect any packet information.

- The Server is an end system that manages one or more TWAMP sessions, and is capable of configuring per-session state in the end-points. However, a Server associated with a Session-Reflector would not have the capability to return the results of a test session, and this is a difference from OWAMP.

- The Fetch-Client entity does not exist in the TWAMP architecture, as the Session-Reflector does not collect any packet information to be fetched. Consequently there is no need for the Fetch-Client.

An example of possible relationship scenarios between these roles are presented below. In this example different logical roles are played on different hosts. Unlabeled links in the figure are unspecified by this document and may be proprietary protocols.

```
+----------------+               +-------------------+
| Session-Sender |<-TWAMP-Test-->| Session-Reflector |
+----------------+               +-------------------+
  ^                                     ^
  |                                     |
  |                                     |
  |                                     |
  |  +----------------+<----------------+
  |  |     Server     |
  |  +----------------+
  |    ^
  |    |
  | TWAMP-Control
  |    |
  v    v
+----------------+
| Control-Client |
+----------------+
```

As in OWAMP [RFC4656], different logical roles can be played by the same host. For example, in the figure above, there could be actually two hosts: one playing the roles of Control-Client and Session-Sender, and the other playing the roles of Server and Session-Reflector. This example is shown below.

```
+-----------------+                   +-------------------+
| Control-Client  |<--TWAMP Control-->|      Server       |
|                 |                   |                   |
| Session-Sender  |<--TWAMP-Test----->| Session-Reflector |
+-----------------+                   +-------------------+
```

Additionally, following the guidelines of OWAMP [RFC4656], TWAMP has been defined to allow for small test packets that would fit inside the payload of a single ATM cell (only in unauthenticated mode).

2. Protocol Overview

The Two-way Active Measurement Protocol is an open protocol for measurement of two-way metrics. It is based on OWAMP [RFC4656] and adheres to its overall architecture and design. The protocol defined in this document extends and changes OWAMP [RFC4656] as follows:

- Define a new logical entity, Session-Reflector, in place of the Session-Receiver.

- Define the Session-Reflector behavior in place of the Session-Receiver behavior of OWAMP [RFC4656].

- Define a new test packet format for packets transmitted from the Session-Reflector to Session-Sender.

- Fetch client does not exist in the TWAMP architecture.

3. TWAMP Control

All TWAMP Control messages are similar in format and follow similar guidelines to those defined in section 3 of OWAMP [RFC4656] with the exceptions outlined in the following sections. All OWAMP [RFC4656] Control messages except for the Fetch-Session command apply to TWAMP.

3.1 Connection Setup

Connection establishment of TWAMP follows the same procedure defined in section 3.1 of OWAMP [RFC4656]. The host that initiates the TCP connection takes the roles of Control-Client and (in the two-host implementation) the Session-Sender. The host that acknowledges the TCP connection accepts the roles of Server and (in the two-host implementation) the Session Reflector.

3.2 Integrity Protection

Integrity protection of TWAMP follows the same procedure defined in section 3.2 of OWAMP [RFC4656].

3.3 Value of the Accept Fields

Accept values used in TWAMP are the same as the values defined in section 3.3 of OWAMP [RFC4656].

3.4 TWAMP Control Commands

TWAMP control commands are as defined in section 3.4 of OWAMP [RFC4656] except that the Fetch-Session command does not apply to TWAMP.

3.5 Creating Test Sessions

Test sessions creation follows the same procedure as defined in section 3.5 of OWAMP [RFC4656].

In order to distinguish the session as a two-way versus a one-way measurement session the first octet of the Request-Session command MUST be set to 5. Value of 5 indicates that this is a Request-Session for a two-way metrics measurement session.

In OWAMP, the Conf-Sender field is set to 1 when the Request-Session message describes a task where the Server will configure a one-way test packet sender. Likewise, the Conf-Receiver field is set to 1 when the message describes the configuration for a Session-Receiver. In TWAMP, both endpoints perform in these roles, with the Session-Sender first sending and then receiving test packets. The Session-Reflector first receives the test packets, and returns each test packet to the Session-Sender as fast as possible.

Both Conf-Sender and Conf-Receiver MUST be set to 0 since the Session-Reflector will both receive and send packets, and the roles are established according to which host initiates the TCP connection for control. The server MUST interpret any non-zero value as zero.

The Session-Reflector in TWAMP does not process incoming test packets for performance metrics and consequently does not need to know the number of incoming packets and their timing schedule. Consequently the Number of Scheduled Slots and Number of Packets MUST be set to 0.

The Sender Port is the UDP port from which TWAMP-Test packets will be sent. Receiver Port is the UDP port to which TWAMP test packets are reflected by the Session-Reflector (the port where the Session-Sender wants to receive test packets).

The Sender Address and Receiver Address fields contain, respectively, the sender and receiver addresses of the endpoints of the Internet path over which a TWAMP test session is requested. They MAY be set to 0, in which case the IP addresses used for the Session-Sender to Session-Reflector Control Message exchange MUST be used in the test packets.

The SID is as defined in OWAMP [RFC4656]. Since the SID is always generated by the receiving side, the Session-Reflector determines the SID, and the SID in the Request-Session message MUST be set to 0.

The Start Time is as as defined in OWAMP [RFC4656].

The Timeout is interpreted differently from the definition in OWAMP [RFC4656]. In TWAMP, Timeout is the interval that the Session-Reflector MUST wait after receiving a Stop-Sessions message. In case there are test packets still in transit, the Session Reflector MUST reflect them if they arrive within the timeout interval following the reception of the Stop-Sessions message. The Session-Reflector MUST NOT reflect packets that are received beyond the timeout.

Type-P descriptor is as defined in OWAMP [RFC4656]. The only capability of this field is to set the Differentiated Services Code Point (DSCP) as defined in [RFC2474]. The same value of DCSP MUST be used in test packets reflected by the Session-Reflector.

Since there are no Schedule Slot Descriptions, the Request-Session Message is completed by MBZ and HMAC fields. This completes one logical message, referred to as the Request-Session Command.

The Session-Reflector MUST respond to each Request-Session Command with an Accept-Message as defined in OWAMP [RFC4656]. The Port is the port to which TWAMP test packets are sent by the Session-Sender toward the Session-Reflector. In other words, the Port field

indicates the port number where the Session-Reflector expects to receive packets from the Session-Sender.

3.6 Send Schedules

The Send Schedule for test packets defined in section 3.6 of OWAMP [RFC4656] is not used in TWAMP. The Control-Client and Session-Sender MAY autonomously decide the Send Schedule. The Session-Reflector SHOULD return each test packet to the Session-Sender as quickly as possible.

3.7 Starting Test Sessions

The procedure and guidelines for Starting test sessions is the same as defined in section 3.7 of OWAMP [RFC4656].

3.8 Stop-Sessions

The procedure and guidelines for Stopping test sessions is the same as defined in section 3.8 of OWAMP [RFC4656]. The Stop-Session command can only be issued by the Session-Sender. The Next SeqNo and Number of Skip Ranges MUST be set to 0 and the message MUST NOT contain any session description records or skip ranges. The message is terminated with a single block HMAC, to complete the Stop-Sessions Command.

3.9 Fetch-Session

The purpose of TWAMP is measurement of two-way metrics. Two-way measurements do not rely on packet level data collected by the Session-Reflector such as sequence number, timestamp, and TTL. As such the protocol does not require the retrieval of packet level data from the Server and the Fetch-Session command is not defined in TWAMP.

4. TWAMP Test

The TWAMP test protocol is similar to the OWAMP [RFC4656] test protocol with the exception that the Session-Reflector transmits test packets to the Session-Sender in response to each test packet it receives. TWAMP defines two different test packet formats, one for packets transmitted by the Session-Sender and one for packets transmitted by the Session-Reflector. As with OWAMP [RFC4656] test protocol there are three modes: unauthenticated, authenticated, and encrypted.

4.1 Sender Behavior

The sender behavior is determined by the configuration of the Session-Sender and is not defined in this standard. Further, the Session-Reflector does not need to know the Session-Sender behaviour to the degree of detail as needed in OWAMP [RFC4656]. Additionally the Session-Sender collects and records the necessary information provided from the packets transmitted by the Session-Reflector for measuring two-way metrics. The information recording based on the received packet by the Session-Sender is implementation dependent.

4.1.1 Packet Timings

Since the Send Schedule is not communicated to the Session-Reflector, there is no need for a standardized computation of packet timing.

Regardless of any scheduling delays, each packet that is actually sent MUST have the best possible approximation of its real time of departure as its timestamp (in the packet).

4.1.2 Packet Format and Content

The Session-Sender packet format and content follow the same procedure and guidelines as defined in section 4.1.2 of OWAMP [RFC4656] (with the exception of the reference to the Send Schedule).

4.2 Reflector Behavior

TWAMP requires the Session-Reflector to transmit a packet to the Session-Sender in response to each packet it receives.

As packets are received the Session-Reflector will,

- Timestamp the received packet. Each packet that is actually received MUST have the best possible approximation of its real time of arrival entered as its timestamp (in the packet).

- In authenticated or encrypted mode, decrypt the first block (16 octets) of the packet body.

- Copy the packet sequence number into the corresponding reflected packet to the Session-Sender.

- Transmit a test packet to the Session-Sender in response to every received packet. The response MUST be generated as immediately as possible. The format and content of the test packet is defined in section 5.2.1. Prior to the transmission of the test packet, the Session-Reflector MUST enter the best possible approximation of its actual sending time of as its Timestamp (in the packet). This permits the determination of the elapsed time between the reception of the packet and its transmission.

- Packets not received within the Timeout are ignored by the Reflector. The Session-Reflector MUST NOT generate a test packet to the Session-Sender for packets that are ignored.

4.2.1 TWAMP-Test Packet Format and Content

The Session-Reflector MUST transmit a packet to the Session-Sender in response to each packet received. The Session-Reflector SHOULD transmit the packets as immediately as possible. The Session-Reflector SHOULD set the TTL in IPV4 (or Hop Limit in IPv6) in the UDP packet to 255.

The test packet will have the necessary information for calculating two-way metrics by the Session-Sender. The format of the test packet depends on the mode being used. The various formats of the packet are presented below.

For unauthenticated mode:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Timestamp                            |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Error Estimate        |           MBZ                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Receive Timestamp                    |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sender Sequence Number                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Sender Timestamp                         |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Sender Error Estimate    |           MBZ                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                         Packet Padding                        .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

For authenticated and encrypted modes:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        MBZ (12 octets)                        |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Timestamp                            |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Error Estimate        |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                        MBZ (6 octets)                         |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Receive Timestamp                      |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        MBZ (8 octets)                         |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                        Sender Sequence Number                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        MBZ (12 octets)                        |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Sender Timestamp                         |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Sender Error Estimate    |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                        MBZ (6 octets)                         |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        HMAC (16 octets)                       |
|                                                               |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                                                               |
.                                                               .
.                         Packet Padding                        .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Sequence Number is the sequence number of the test packet according to its arrival at the Session-Reflector. It starts with zero and is incremented by one for each subsequent packet. The Sequence Number generated by the Session-Reflector is independent from the sequence number of the arriving packets.

Timestamp and Error Estimate are the Session-Reflector's transmit timestamp and error estimate for the reflected test packet, respectively. The format of all timestamp and error estimate fields follow the definition and formats defined by OWAMP[RFC4656].

Sender Timestamp and Sender Error Estimate are exact copies of the timestamp and error estimate from the Session-Sender test packet that corresponds to this test packet.

Receive Timestamp is the time the test packet was received by the reflector. The difference between Timestamp and Receive Timestamp is the amount of time the packet was in transition in the Session-Reflector. The Error Estimate associated with the Timestamp field also applies to the Receive Timestamp.

Sender Sequence Number is a copy of the Sequence Number of the packet transmitted by the Session-Sender that caused the Session-Reflector to generate and send this test packet.

Similar to OWAMP [RFC4656] the TWAMP packet layout is the same in authenticated and encrypted modes. The encryption operation of Session-Sender packet follow the same rules of Session-Sender packets as defined in OWAMP [RFC4656].

The minimum data segment length is, therefore, 40 octets in unauthenticated mode, and 80 octets in both authenticated mode and encrypted modes (with the implication that the later two modes will not fit in a single ATM cell).

The Session-Reflector TWAMP-Test packet layout is the same in authenticated and encrypted modes. The encryption operations are, however, different. The difference is that in encrypted mode both the sequence numbers and timestamps are encrypted to provide maximum data integrity protection while in authenticated mode the sequence numbers are encrypted and the timestamps are sent in clear text. Sending the timestamp in clear text in authenticated mode allows one to reduce the time between when a timestamp is obtained by a reflector and when the packet is reflected out. In encrypted mode, both the sender and reflector have to fetch the timestamp, encrypt it, and send it; in authenticated mode, the middle step is

removed, potentially improving accuracy (the sequence number can be encrypted before the timestamp is fetched).

In authenticated mode, the first block (16 octets) of each packet is encrypted using AES Electronic Cookbook (ECB) mode.

Obtaining the key, encryption method, and packet padding follows the same procedure as OWAMP as described below.
Similarly to each TWAMP-Control session, each TWAMP-Test session has two keys: an AES Session-key and an HMAC Session-key. However, there is a difference in how the keys are obtained: in the case of TWAMP-Control, the keys are generated by the client and communicated (as part of the Token) during connection setup as part of Set-Up-Response message; in the case of TWAMP-Test, described here, the keys are derived from the TWAMP-Control keys and the SID.

The TWAMP-Test AES Session-key is obtained as follows: the TWAMP-Control AES Session-key (the same AES Session-key as is used for the corresponding TWAMP-Control session, where it is used in a different chaining mode) is encrypted, using AES, with the 16-octet session identifier (SID) as the key; this is a single-block ECB encryption; its result is the TWAMP-Test AES Session-key to use in encrypting (and decrypting) the packets of the particular TWAMP-Test session. Note that all of TWAMP-Test AES Session-key, TWAMP-Control AES Session-key, and the SID are comprised of 16 octets.

The TWAMP-Test HMAC Session-key is obtained as follows: the TWAMP-Control HMAC Session-key (the same HMAC Session-key as is used for the corresponding TWAMP-Control session) is encrypted, using AES, with the 16-octet session identifier (SID) as the key; this is a two-block CBC encryption, always performed with IV=0; its result is the TWAMP-Test HMAC Session-key to use in authenticating the packets of the particular TWAMP-Test session. Note that all of TWAMP-Test HMAC Session-key and TWAMP-Control HMAC Session-key are comprised of 32 octets, while the SID is 16 octets.

ECB mode used for encrypting the first block of TWAMP-Test packets in authenticated mode does not involve any actual chaining; this way, lost, duplicated, or reordered packets do not cause problems with deciphering any packet in an TWAMP-Test session.

In encrypted mode, the first two blocks (32 octets) are encrypted using AES CBC mode. The AES Session-key to use is obtained in the same way as the key for authenticated mode. Each TWAMP-Test packet is encrypted as a separate stream, with just one chaining operation; chaining does not span multiple packets so that lost, duplicated, or reordered packets do not cause problems. The

initialization vector for the CBC encryption is a value with all bits equal to zero.

Implementation note: Naturally, the key schedule for each TWAMP-Test session MAY be set up only once per session, not once per packet.

HMAC in TWAMP-Test only covers the part of the packet that is also encrypted. So, in authenticated mode, HMAC covers the first block (16 octets); in encrypted mode, HMAC covers two first blocks (32 octets). In TWAMP-Test HMAC is not encrypted (note that this is different from TWAMP-Control, where encryption in stream mode is used, so everything including the HMAC blocks ends up being encrypted).

In unauthenticated mode, no encryption or authentication is applied.

Packet Padding in TWAMP-Test SHOULD be pseudo-random (it MUST be generated independently of any other pseudo-random numbers mentioned in this document). However, implementations MUST provide a configuration parameter, an option, or a different means of making Packet Padding consist of all zeros.

5. Implementers Guide

This section serves as guidance to implementers of TWAMP. Two architectures are presented in this section for implementations where two hosts play the subsystem roles of TWAMP. Although only two architectures are presented here the protocol does not require their use. Similar to OWAMP [RFC4656] TWAMP is designed with complete flexibility to allow different architectures that suite multiple system requirements.

5.1 Complete TWAMP

In this example the roles of Control-Client and Session-Sender are implemented in one host referred to as the controller and the roles of Server and Session-Reflector are implemented in another host referred to as the responder.

```
        controller                                responder
    +-----------------+                  +------------------+
    | Control-Client  |<--TWAMP-Control-->| Server           |
```

```
          |                 |                   |                    |
          | Session-Sender  |<--TWAMP-Test----->| Session-Reflector  |
          +-----------------+                   +--------------------+
```

This example provides an architecture that supports the full TWAMP standard. The controller establishes the test session with the responder through the TWAMP-Control protocol. After the session is established the controller transmits test packets to the responder. The responder follows the Session-Reflctor behavior of TWAMP as described in section 4.2.

5.2 TWAMP Light

In this example the roles of Control-Client, Server, and Session-Sender are implemented in one host referred to as the controller and the role of Session-Reflector is implemented in another host referred to as the responder.

```
              controller                                responder
          +-----------------+                   +--------------------+
          |     Server      |<----------------->|                    |
          | Control-Client  |                   | Session-Reflector  |
          | Session-Sender  |<--TWAMP-Test----->|                    |
          +-----------------+                   +--------------------+
```

This example provides a simple architecture for responders where their role will be to simply act as light test points in the network. The controller establishes the test session with the Server through non-standard means. After the session is established the controller transmits test packets to the responder. The responder follows the Session-Reflector behavior of TWAMP as described in section 4.2 with the following exceptions. The Session-Reflector SHOULD copy the sequence number of the received packet to the Sequence Number field of the reflected packet. This is necessary since in case of TWAMP Light the Session-Reflector does not have knowledge of the session state. The controller receives the reflected test packets and collects two-way metrics. This architecture allows for collection of two-way metrics.

This example eliminates the need for the TWAMP-Control protocol and assumes that the Session-Reflector is configured and communicates its configuration with the Server through non-standard means. The Session-Reflector simply reflects the incoming packets back to the

controller while copying the necessary information and generating sequence number and timestamp values per section 5.2.1.

6. Security Considerations

Fundamentally TWAMP and OWAMP use the same protocol for establishment of Control and Test procedures. The main difference between TWAMP and OWAMP is the Session-Reflector behavior in TWAMP vs. the Session-Receiver behavior in OWAMP. This difference in behavior does not introduce any known security vulnerabilities that are not already addressed by the security features of OWAMP. The entire security considerations of OWAMP [RFC4656] applies to TWAMP.

7. Acknowledgements

We would like to thank Nagarjuna Venna, Sharee McNab, Nick Kinraid, and Stanislav Shalunov for their comments, suggestions, reviews, helpful discussion and proof-reading.

8. IANA Considerations

IANA has allocated a well-known TCP port number (861) for the OWAMP-Control part of the OWAMP [RFC4656] protocol which also applies to the TWAMP-Control part of the TWAMP protocol.

9. Internationalization Considerations

The protocol does not carry any information in a natural language, with the possible exception of the KeyID in TWAMP-Control, which is encoded in UTF-8.

10. References

10.1 Normative References

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., Zekauskas, M., "A One-way Active Measurement Protocol (OWAMP)", draft-ietf-ippm-owdp-11.txt, October 2004.

[RFC2681] Almes, G., Kalidindi, S., Zekauskas, M., "A Round-Trip Delay Metric for IPPM". RFC 2681, STD 1, September 1999.

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

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

Authors' Addresses

Kaynam Hedayat
Brix Networks
285 Mill Road
Chelmsford, MA 01824
USA

EMail: khedayat@brixnet.com
URI: http://www.brixnet.com/

Roman M. Krzanowski, Ph.D.
Verizon
500 Westchester Ave.
White Plains, NY
USA

EMail: roman.krzanowski@verizon.com
URI: http://www.verizon.com/

Al Morton
AT&T Labs
Room D3 - 3C06
200 Laurel Ave. South
Middletown, NJ 07748
USA

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

Kiho Yum
Juniper Networks
1194 Mathilda Ave.
Sunnyvale, CA
USA

EMail: kyum@juniper.net
URI: http://www.juniper.com/

Jozef Z. Babiarz
Nortel Networks
3500 Carling Avenue
Ottawa, Ont K2H 8E9
Canada

Email: babiarz@nortel.com
URI: http://www.nortel.com/

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).