

Internet Draft
Expires: May 2008

K. Hedayat
Brix Networks
R. Krzanowski
Verizon
K. Yum
Juniper Networks
A. Morton
AT&T Labs
J. Babiarz
Nortel Networks
November 2007

**A Two-way Active Measurement Protocol (TWAMP)
draft-ietf-ippm-twamp-05**

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

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

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

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

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

Copyright Notice

Copyright (C) The IETF Trust (2007).

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.....	3
1.1	Relationship of Test and Control Protocols.....	3
1.2	Logical Model.....	3
2.	Protocol Overview.....	5
3.	TWAMP Control.....	5
3.1	Connection Setup.....	5
3.2	Integrity Protection.....	6
3.3	Value of the Accept Fields.....	6
3.4	TWAMP Control Commands.....	6
3.5	Creating Test Sessions.....	6
3.6	Send Schedules.....	8
3.7	Starting Test Sessions.....	8
3.8	Stop-Sessions.....	9
3.9	Fetch-Session.....	9
4.	TWAMP Test.....	9
4.1	Sender Behavior.....	9
4.2	Reflector Behavior.....	10
5.	Implementers Guide.....	16
5.1	Complete TWAMP.....	17
5.2	TWAMP Light.....	17
6.	Security Considerations.....	18
7.	Acknowledgements.....	18
8.	IANA Considerations.....	19
9.	Internationalization Considerations.....	20
10.	References.....	21
10.1	Normative References.....	21

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](#)].

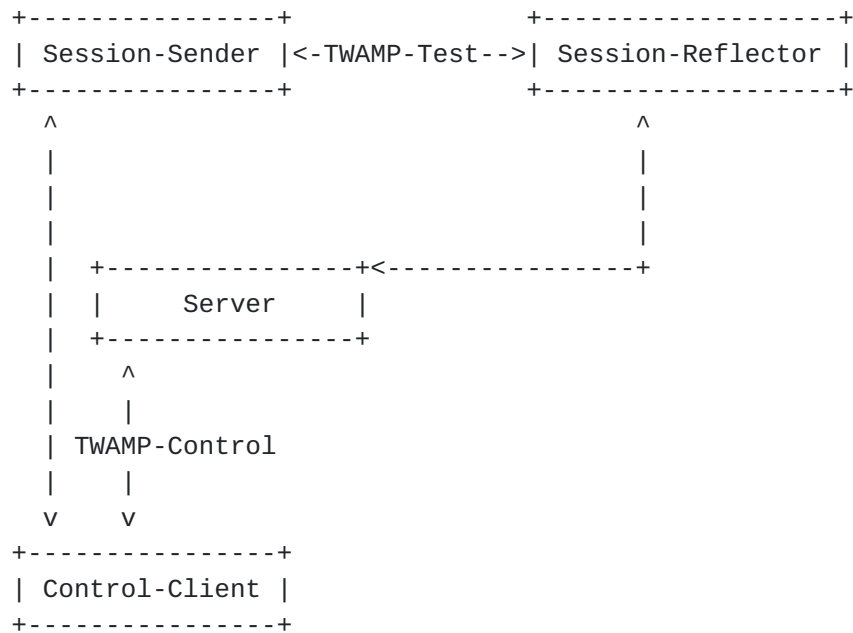
1.2 Logical Model

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

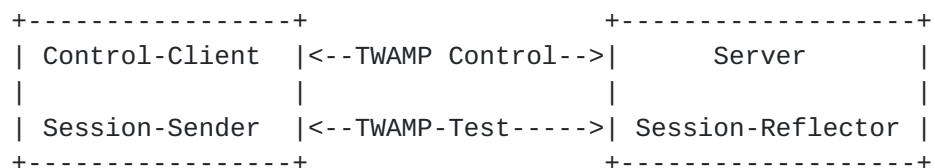
- The Session-Receiver is called the Session-Reflector in the TWAMP architecture. The Session-Reflector has the capability to create and send a measurement packet when it receives a measurement packet. Unlike the Session-Receiver, the Session-Reflector does not collect any packet information.

- 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.



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.



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.

All multi-octet quantities defined in this document are represented as unsigned integers in network byte order unless specified otherwise.

3. TWAMP Control

TWAMP-Control is a derivative of the OWAMP-Control for two-way measurements. 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 mode values are

identical to OWAMP. The only exception is the well-known port number for TWAMP-control. A client opens a TCP connection to the server on well-known port N (Refer to the IANA Considerations section below for the TWAMP-control port number assignment). 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 TWAMP, the first octet is referred to as the Command Number, and the Command Number is a recognized extension mechanism. Readers are encouraged to consult the TWAMP Command Number Registry to determine if there have been additional values assigned.

If a TWAMP server receives an unexpected command number, it MUST respond with the Accept field set to 3 (meaning "Some aspect of request is not supported") in the Server-Start message.

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 and the port to which TWAMP-Test packets will be sent by the Session-Reflector (Session-Sender will use the same UDP port to send and receive packets). Receiver Port is the desired UDP port to which TWAMP test packets will be sent by the Session-Sender (the port where the Session-Reflector is asked to receive test packets). Receiver Port is also the UDP port from which TWAMP test packets will be sent by the Session-Reflector (Session-Reflector will use the same UDP port to send and receive 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]. When the Accept Field = 0, the Port field confirms (repeats) 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.

When the requested Receiver Port is not available (e.g., port in use), the Server at the Session-Reflector MAY suggest an alternate and available port for this session in the Port Field. The Session-Sender either accepts the alternate port, or composes a new Session-Request message with suitable parameters. Otherwise, the Server at the Session-Reflector uses the Accept Field to convey other forms of session rejection or failure and MUST NOT suggest an alternate port. In this case the Port Field MUST be set to zero.

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.
- Sender TTL value is extracted from the TTL/Hop Limit value of received packets. Session-Reflector Implementations **SHOULD** fetch the TTL/Hop Limit value from the IP header of the packet, replacing the value of 255 set by the Session-Sender. If an

[illegible]


```

|                               Sender Sequence Number                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Sender Timestamp                                   |
|                                                                                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Sender Error Estimate   |                               MBZ                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Sender TTL   |                                                                                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                                                                   |
.                                                                                   .
.                               Packet Padding                                   .
.                                                                                   .
|                                                                                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```



```

|                                                                    |
.                                                                    .
.          Packet Padding                                          .
.                                                                    .
|                                                                    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Sequence Number is the sequence number of the test packet according to its transmit order. 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.

Sender TTL is 255 when transmitted by the Session Sender. Sender TTL is set to the Time To Live (or Hop Count) value of the received packet from the IP packet header when transmitted by the Session Reflector.

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, 41 octets in unauthenticated mode, and 104 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 six blocks (96octets) 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.

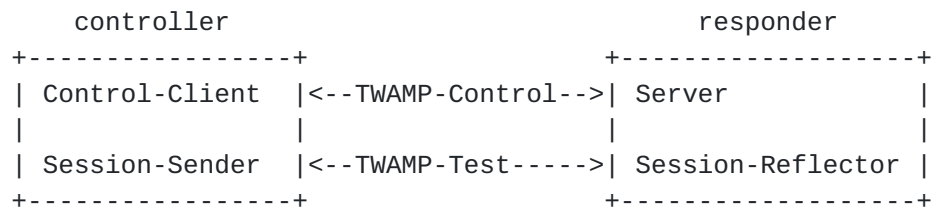
Packet Padding in TWAMP-Test SHOULD be pseudo-random (it MUST be generated independently of any other pseudo-random numbers mentioned in this document). However, implementations MUST provide a configuration parameter, an option, or a different means of making Packet Padding consist of all zeros.

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

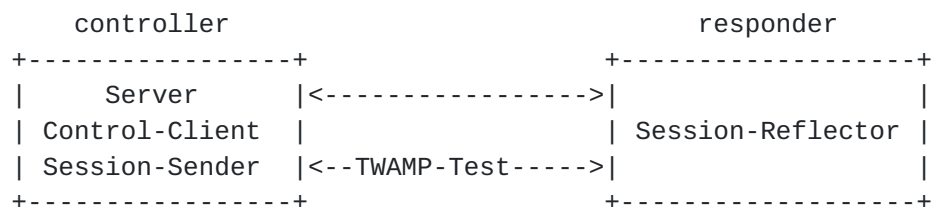
In this example the roles of Control-Client and Session-Sender are implemented in one host referred to as the controller and the roles of Server and Session-Reflector are implemented in another host referred to as the responder.



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-Reflector 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.



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.

In the case of TWAMP Light, the Session-Reflector does not necessarily have knowledge of the session state. IF the Session-Reflector does not have knowledge of the session state, THEN the Session-Reflector MUST copy the Sequence Number of the received packet to the Sequence Number field of the reflected packet. 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 4.2.1](#).

[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.

The only area where TWAMP may introduce new security considerations is the TWAMP Light version described above. The non-standard means to control the responder and establish test sessions SHOULD offer the features listed below.

The non-standard responder control protocol SHOULD have an authenticated mode of operation. The responder SHOULD be configurable to accept only authenticated control sessions.

The non-standard responder control protocol SHOULD have a means to activate the authenticated and encrypted modes of the TWAMP-Test protocol.

[7. Acknowledgements](#)

We would like to thank Nagarjuna Venna, Sharee McNab, Nick Kinraid, Stanislav Shalunov, Matt Zekauskas, Walt Steverson and Jeff Boote

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.

```
...
owamp-control    861/tcp    OWAMP-Control
owamp-control    861/udp    OWAMP-Control
#                [RFC4656]
#                862-872    Unassigned
```

IANA is requested to allocate a well-known TCP/UDP port number for the TWAMP-Control protocol. It would be ideal if the port number assignment was adjacent to the OWAMP assignment. The recommended Keyword for this entry is "twamp-control" and the Description is "Two-way Active Measurement Protocol (TWAMP) Control".

During final editing, port N in [section 3.1](#) should be replaced with the assigned port number.

Since TWAMP adds an additional Control command to the OWAMP-Control specification, and describes behavior when this control command is used, this memo requests creation an IANA registry for the TWAMP Command Number field. The field is not explicitly named in [\[RFC4656\]](#) but is called out for each command. This field is a recognized extension mechanism for TWAMP.

8.1 Registry Specification

IANA will create an TWAMP-Control Command registry. TWAMP-Control commands are specified by the first octet in OWAMP-Control messages as shown in [section 3.4 of \[RFC4656\]](#), and modified by this document. Thus this registry may contain sixteen possible values.

8.2 Registry Management

Because the registry may only contain sixteen values, and because OWAMP and TWAMP are IETF protocols, this registry must only be updated by "IETF Consensus" as specified in [\[RFC2434\]](#) -- an RFC documenting the use that is approved by the IESG. We expect that

new values will be assigned as monotonically increasing integers in the range [0-15], unless there is a good reason to do otherwise.

8.3 Experimental Numbers

[RFC3692] recommends allocating an appropriate number of values for experimentation and testing. It is not clear to the authors exactly how many might be useful in this space, nor if it would be useful that they were easily distinguishable or at the "high end" of the number range. Two might be useful, say one for session control, and one for session fetch. On the other hand, a single number would allow for unlimited extension, because the format of the rest of the message could be tailored, with allocation of other numbers done once usefulness has been proven. Thus, this document will allocate one number, the next sequential number 6, as designated for experimentation and testing.

8.4 Initial Registry Contents

TWAMP-Control Command Registry

Value	Description	Semantics Definition
0	Reserved	
1	Forbidden	
2	Start-Sessions	RFC4656, Section 3.7
3	Stop-Sessions	RFC4656, Section 3.8
4	Fetch-Session	RFC4656, Section 3.9
5	Request-TW-Session	this document, Section 3.5
6	Experimentation	undefined, see Section 8.3 .

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.
- [RFC2434] Narten, T., Alvestrand, H., Guidelines for Writing an IANA Considerations Section in RFCs, [RFC 2474](#), October 1998.

10.2 Informative References

- [RFC3692] Narten, T., Assigning Experimental and Testing Numbers Considered Useful, [RFC 3692](#), January 2004.

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

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

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

