IPPM WG
Internet-Draft
Intended status: Standards Track
Expires: June 26, 2017

R. Civil
Ciena Corporation
A. Morton
AT&T Labs
R. Rahman
M. Jethanandani
Cisco Systems
K. Pentikousis, Ed.
Travelping
L. Zheng
Huawei Technologies
December 23, 2016

Two-Way Active Measurement Protocol (TWAMP) Data Model
draft-ietf-ippm-twamp-yang-02

Abstract

This document specifies a data model for client and server implementations of the Two-Way Active Measurement Protocol (TWAMP). We define the TWAMP data model through Unified Modeling Language (UML) class diagrams and formally specify it using YANG.

Status of This Memo

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

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

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

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

1.2. Terminology

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

1.3. Document Organization

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

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

2. Scope, Model, and Applicability

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

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

As per [RFC5357], unlabeled links in Figure 1 are left unspecified and may be proprietary protocols.

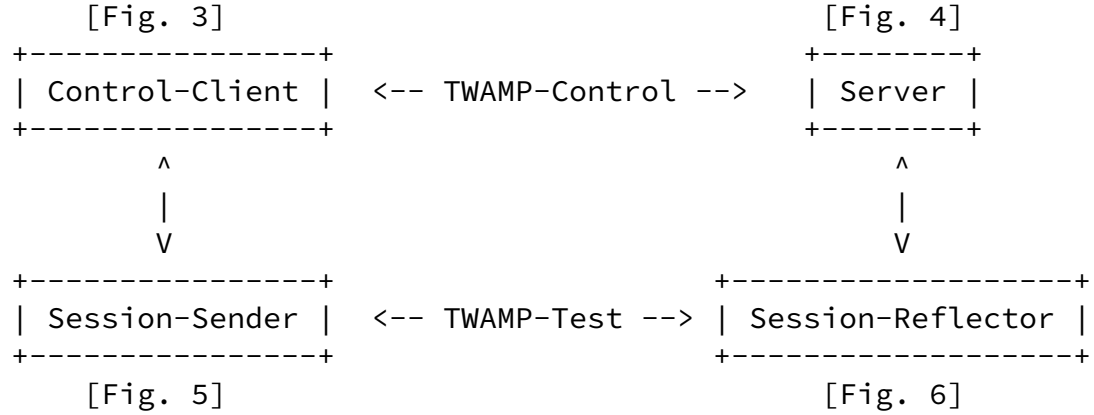


Figure 1: Annotated TWAMP logical model

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].

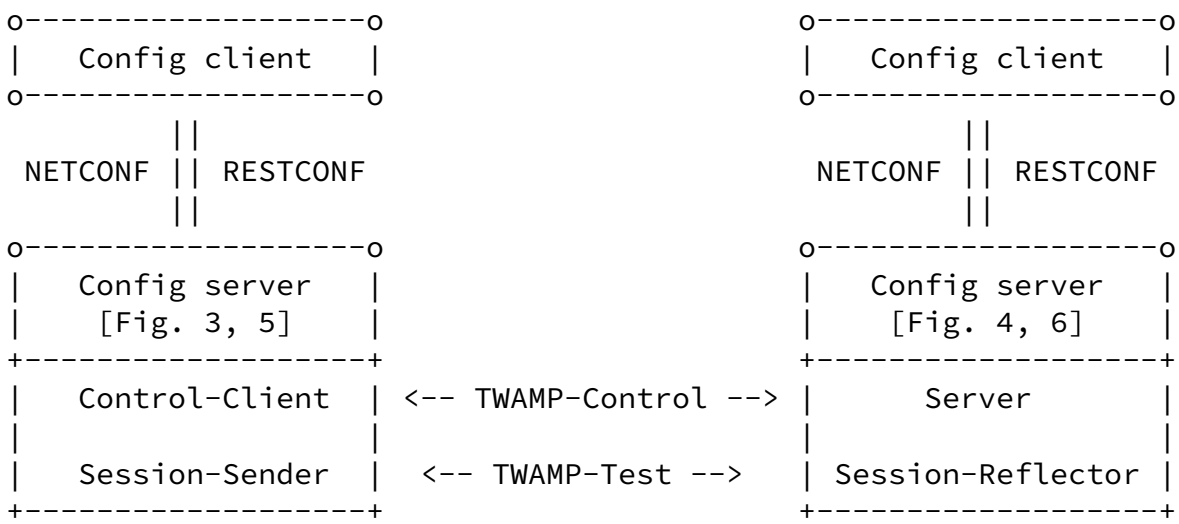


Figure 2: Simplified TWAMP model and protocols

We note that 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 perfmormance measurement results are retrieved, or how stored results are cleared, and so on, are not addressed by the configuration model defined in this docuemnt. 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.

Global configuration items relate to parameters that are set on a per device level. For example, the administrative status of the device with respect to whether it allows TWAMP sessions and, if so, in what capacity (e.g. Control-Client, Server or both), are typical instances of global configuration items.

A second category includes attributes that can be configured on a per 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:

o A name which can be used to uniquely identify at the Control-

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

- 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 parameter (Section 4.2 of [RFC5357] can be configured.

Read-only access to other data model parameters, such as the Sender

IP address is foreseen. Each test session can be uniquely identified by the 4-tuple mentioned in Section 3.2.

4. Data Model Parameters

This section defines the TWAMP data model using UML and 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.

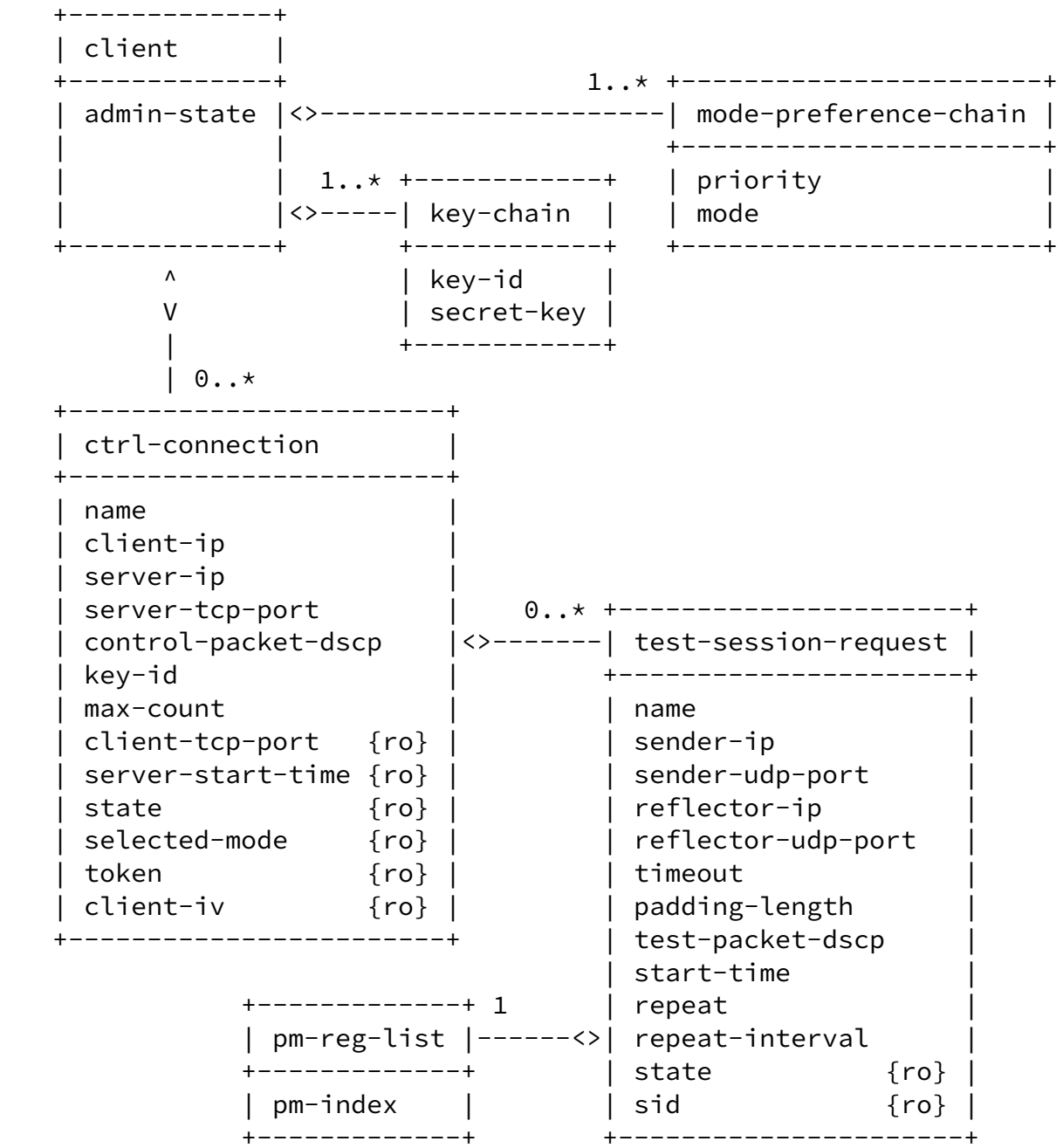


Figure 3: TWAMP Control-Client UML class diagram

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 each priority (expressed as a 16-bit unsigned integer, where zero is the highest priority and subsequent values monotonically increasing) with their corresponding mode (expressed as a 32-bit Hexadecimal value).

Depending on the Modes available in the Server Greeting, the Control-Client MUST choose the highest priority Mode from the configured mode-preference-chain list.

Note that the list of preferred Modes may set bit position combinations when necessary, such as when referring to the extended TWAMP features in [RFC5618], [RFC5938], [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 128-bits for the AES Session-key used for encryption and a 256-bit HMAC-SHA1 Session-key used for authentication (see Section 6.10 of [RFC4656]).

Each client container also holds a list of ctrl-connections, where each item in the list describes a TWAMP control connection that will be initiated by this Control-Client. There SHALL be one instance of 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 that is 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.

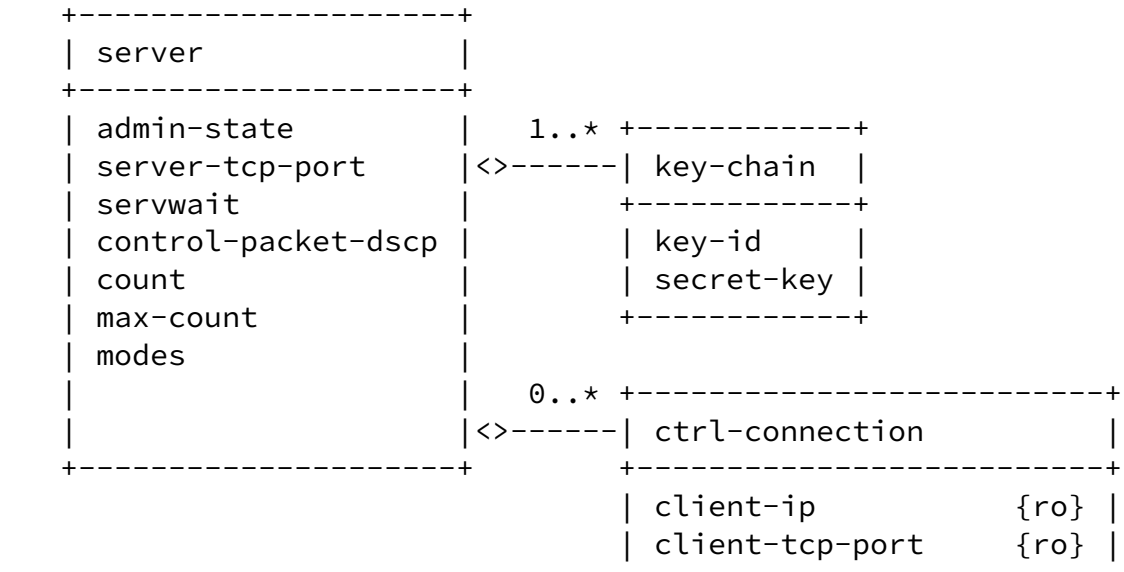
The Control-Client is also responsible for scheduling TWAMP-Test sessions and collecting the respective results, so test-session-request holds information related to these actions (e.g. pm-index, repeat-interval).

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 are applied to all incoming TWAMP-Control connections.



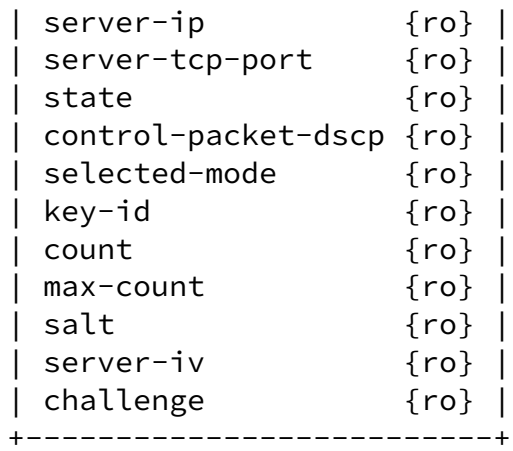


Figure 4: TWAMP Server UML class diagram

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.

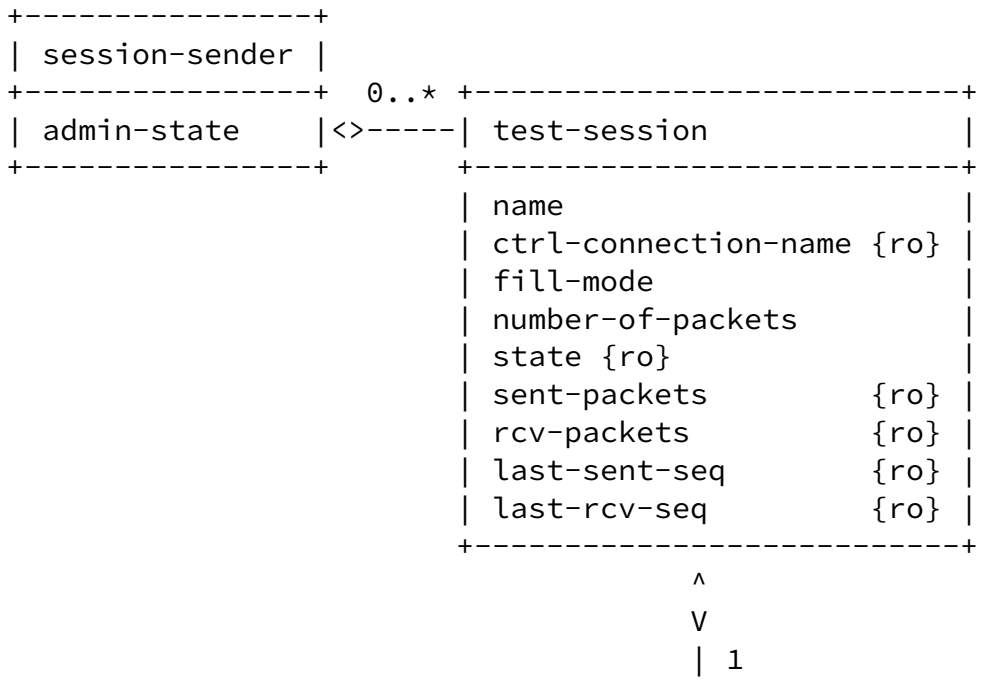
Each incoming control connection that is active on the Server will be represented by an instance of a ctrl-connection object. There SHALL be one instance of ctrl-connection per incoming TWAMP-Control (TCP) connection that is received and active on the Server device.

All items in the ctrl-connection object are read-only. Each instance of ctrl-connection can be uniquely identified by the 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port}.

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.



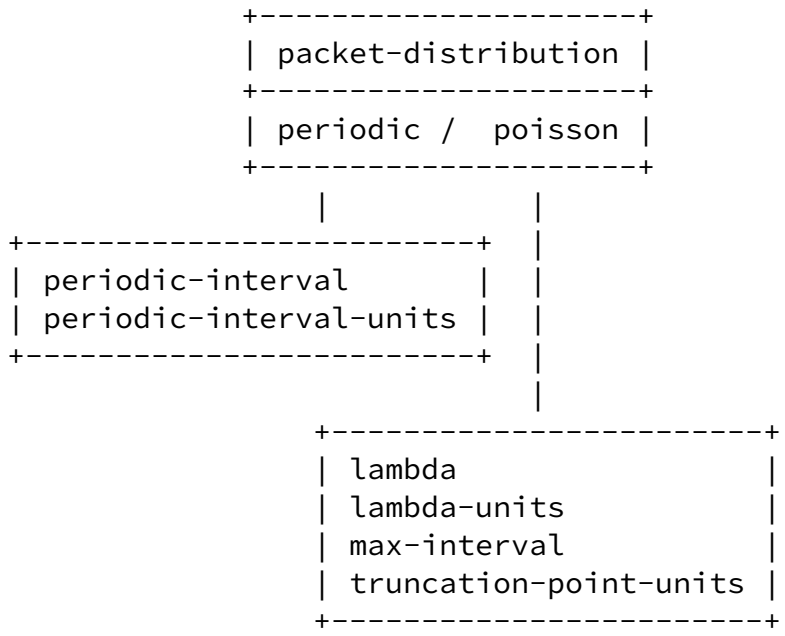


Figure 5: TWAMP Session-Sender UML class diagram

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.

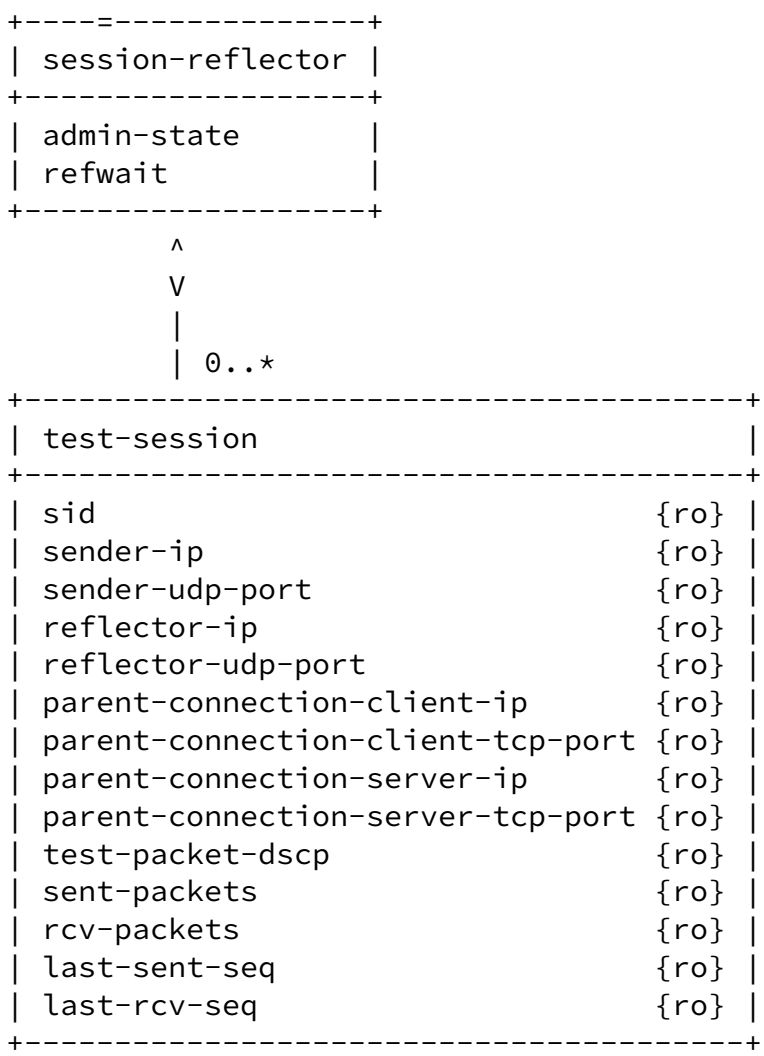


Figure 6: TWAMP Session-Reflector UML class diagram

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 the server/ctrl-connection object. This 4-tuple allows the user to trace back from the TWAMP-Test session to the (parent) TWAMP-Control connection that negotiated this test session.

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        string
```

```
|  |  +--rw secret-key?   string
|  +--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?                uint32
|     +--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?        dynamic-port-number
|        +--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?                 uint32
|        +--rw repeat-interval?        uint32
|        +--rw pm-reg-list* [pm-index]
|        |  +--rw pm-index    uint16
|        +--ro state?                  test-session-state
|        +--ro sid?                    string
+--rw server! {server}?
|  +--rw admin-state             boolean
|  +--rw server-tcp-port?        inet:port-number
|  +--rw servwait?               uint32
|  +--rw control-packet-dscp?    inet:dscp
|  +--rw count?                  uint32
|  +--rw max-count?              uint32
|  +--rw modes?                  twamp-modes
|  +--rw key-chain* [key-id]
|  |  +--rw key-id        string
|  |  +--rw secret-key?   string
|  +--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?                  uint32
|     +--ro max-count?              uint32
|     +--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?          uint32
|     |  |  +--rw periodic-interval-units?    time-units
|     |  +--:(poisson)
|     |     +--rw lambda?                    uint32
|     |     +--rw lambda-units?              uint32
|     |     +--rw max-interval?              uint32
|     |     +--rw truncation-point-units?    time-units
|     +--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
```

5.2. YANG Module

This section presents the YANG module for the TWAMP data model defined in this document.

```
<CODE BEGINS> file "2016-12-23"

module ietf-twamp {
  //namespace need to be assigned by IANA
  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:
      Control-Client, Server, Session-Sender, and Session-Reflector.
      See Fig. 1 of draft-ietf-ippm-twamp-yang for an illustration
      of the annotated TWAMP logical model.

      The YANG module uses features to indicate which of the four
      logical entities are supported by an implementation.";
```



```
    revision 2016-07-07 {
      description
          "Revision appearing in draft-ietf-ippm-twamp-yang-01.
          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. See RFC 7717 Section 7.";
        }
        bit authenticated {
          position 1;
          description
              "Authenticated mode. See RFC 7717 Section 7
```

```
        and RFC 4656 Section 6.";
      }
      bit encrypted {
        position 2;
        description
          "Encrypted mode. See RFC 7717 Section 7 and
          RFC 4656 Section 6.";
      }
      bit unauth-test-encrpyt-control {
        position 3;
        description
          "Mixed Security Mode: TWAMP-Test protocol security
          mode in Unauthenticated mode, TWAMP-Control protocol
          in Encrypted mode.";
        reference
          "RFC 5618: Mixed Security Mode for the Two-Way Active
          Measurement Protocol (TWAMP)";
      }
      bit individual-session-control {
        position 4;
        description
          "Individual Session Control.";
```



```
        reference
          "RFC 5938: Individual Session Control Feature
          for the Two-Way Active Measurement Protocol (TWAMP)";
      }
      bit reflect-octets {
        position 5;
        description
          "Reflect Octets Capability.";
        reference
          "RFC 6038: Two-Way Active Measurement Protocol (TWAMP)
          Reflect Octets and Symmetrical Size Features";
      }
      bit symmetrical-size {
        position 6;
        description
          "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
              "IKEv2Derived Mode Capability.";
          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 used during a
          TWAMP-Control Connection setup between a Control-Client
          and a Server. RFC 7717 Section 7 summarizes the
          TWAMP-Modes registry.";
    }

    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 "Control-Client control connection state";
    }

    typedef test-session-state {
      type enumeration {
        enum accepted {
          value 0;
          description
              "Indicates that the TWAMP-Test session request
              is accepted.";
        }
        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 "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 according to RFC 5357
          (Section 3.1): [a] Server MAY discontinue any established
          control connection when no packet associated with that
          connection has been received within SERVWAIT seconds.";
        }
      }
      description "Server 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 "Session-Sender session state.";
    }

    typedef padding-fill-mode {
      type enumeration {
        enum zero {
          description "Packets will be padded with
          all zeros";
        }
        enum random {
          description "Packets will be padded with
          pseudo-random numbers";
        }
      }
      description "Indicates what type of packet padding is
      to be used for the UDP TWAMP-Test packets.";
    }

    typedef time-units  {
      type enumeration {
        enum s {
          description "Seconds.";
```

```
    }
    enum ms {
      description "Milliseconds.";
    }
    enum us {
      description "Microseconds.";
    }
    enum ns {
      description "Nanoseconds.";
    }
  }
  description "TWAMP configuration parameters time units.";
}

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.";
}

feature server {
  description
      "Indicates that the device supports configuration
      of the TWAMP Server.";
}

feature session-sender {
  description
      "Indicates that the device supports configuration
      of the TWAMP Session-Sender.";
}

feature session-reflector {
  description
      "Indicates that the device supports configuration
      of the TWAMP Session-Reflector.";
}
```

```
    /*
     * Reusable node groups
     */

    grouping key-management {

      list key-chain {
          key key-id;
          leaf key-id {
            type string {
              length 1..80;
            }
            description
              "KeyID to be used for a TWAMP-Control connection.";
          }

          leaf secret-key {
            type string;
            description
              "The corresponding secret key for the
              TWAMP-Control connection.";
          }
          description
              "Relates KeyIDs with the respective secret keys
              for a TWAMP-Control connection.";
      }
      description "TWAMP-Control key management.";
    }

    grouping maintenance-statistics {

      leaf sent-packets {
        type uint32;
        config false;
        description "Packets sent";
      }

      leaf rcv-packets {
        type uint32;
        config false;
        description "Packets received";
      }
```

```
      leaf last-sent-seq {
        type uint32;
        config false;
        description "Last sent sequence number";
      }
```



```
      leaf last-rcv-seq {
        type uint32;
        config false;
        description "Last received sequence number";
      }
      description "TWAMP-Test maintenance statistics";
    }


    /*
     * Configuration data nodes
     */

    container twamp {
      description
          "TWAMP logical entity configuration grouping.";

      container client {
        if-feature control-client;
        presence client;
        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 "Priority.";
        }
        leaf mode {
          type twamp-modes;
          description "Supported TWAMP Mode.";

        }
        description
            "Indicates the preferred order of use for the
            corresponding supported TWAMP Modes";
      }
```



```
      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 belonging to 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 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
         "The KeyID value that is selected
          for this TWAMP-Control connection.";

      }

      leaf max-count {
        type uint32 {
```

```
            range 1024..4294967295;
          }
          default 32768;
          description
            "This parameter limits the maximum Count value.

            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.";
        }

        leaf client-tcp-port {
          type inet:port-number;
          config false;
          description
            "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
            "The Start-Time advertized by the Server in the
            Server-Start message (RFC 4656, Section 3.1). This is
            a timestamp representing the time when the current
```



```
            instantiation of the Server started operating.";
        }

        leaf state {
          type control-client-connection-state;
          config false;
          description
            "Indicates the currest 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.

              AES Session-key and HMAC Session-key are generated
              randomly by the Control-Client. AES Session-key and
              HMAC Session-key MUST be generated with sufficient
              entropy not to reduce the security of the underlying
              cipher. The token itself is encrypted
              using the AES (Advanced Encryption Standard) in
              Cipher Block Chaining (CBC). Encryption MUST be
              performed using an Initialization Vector (IV)
              of zero and a key derived from the shared secret
              associated with KeyID. Challenge is the same as
              transmitted by the Server in the clear; see also the
              last paragraph of Section 6 in RFC 4656.";
            reference
              "RFC 4086: Randomness Requirements for Security";
          }

          leaf client-iv {
```



```
            type binary {
              length 16;
            }
            config false;
            description
              "The Control-Client Initialization Vector (Client-IV)
              is generated randomly by the Control-Client.
```

```
            Client-IV merely needs to be unique (i.e., it MUST
            never be repeated for different sessions using the
            same secret key; a simple way to achieve that without
            the use of cumbersome state is to generate the
            Client-IV values using a cryptographically secure
            pseudo-random number source.";
          }

          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 dynamic-port-number;
              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.
                  A value of zero indicates that the Control-Client
```

```
        SHALL auto-allocate a UDP port number for this
        TWAMP-Test session. The configured
        (or auto-allocated) value is advertized in the
        Sender Port field of the Request-TW-session message
        (see also Section 3.5 of 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. If this value is not set, the device SHALL
          use the same port number as defined in the
          server-tcp-port parameter of this
          test-session-request's parent
          twamp/client/ctrl-connection.";
    }

    leaf timeout {
      type uint64;
      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;
          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 RFC 5357, Section 3.4).

            The start-time value is placed in the Start Time
            field of the Request-TW-Session message.

            The default value of 0 indicates that the session
```

```
                will be started as soon as the Start-Sessions message
                is received.";
            }

            leaf repeat {
```



```
              type uint32;
              default 0;
              description
                  "This value determines if the TWAMP-Test session must
                  be repeated. When a test session has completed, the
                  repeat parameter is checked.

                  The value of 0 indicates that the session MUST NOT be
                  repeated.

                  If the 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.

                  The value of 4,294,967,295 indicates that the test
                  session SHALL be repeated *forever* using the
                  information in repeat-interval parameter, and
                  SHALL NOT decrement the value.";
            }

            leaf repeat-interval  {
              when "../repeat!='0'" {
                description
                  "This parameter determines the timing of repeated
                  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 minutes specified in the
```

```
            repeat-interval parameter before negotiating the
            new instance of this TWAMP-Test session.";
        }
        type uint32;
        default 0;
        description "Repeat interval (in minutes)";
      }

      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 server;
      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;
          }
          default 900;
          description
              "TWAMP-Control (TCP) session timeout, in seconds
              (RFC 5357, Section 3.1)).";
        }

        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.";
        }

        leaf count {
          type uint32 {
            range 1024..4294967295;
          }
          description
              "Parameter used in deriving a key from a shared
              secret as described in Section 3.1 of  RFC 4656,
              and are communicated to the Control-Client as part
              of the Server Greeting message.
```

```
            count MUST be a power of 2.

            count MUST be at least 1024.

            count SHOULD be increased as more computing power
            becomes common.";
        }
        leaf max-count {
          type uint32 {
            range 1024..4294967295;
          }
          default 32768;
          description
              "This parameter limits the maximum Count value.

              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.";
        }

        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.";
          }

          leaf count {
            type uint32 {
              range 1024..4294967295;
            }
            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.";
```

```
          }

          leaf max-count {
            type uint32 {
              range 1024..4294967295;
            }
            description
              "The max-count value that is in use by this
              TWAMP-Control connection. This will usually be the
              same value as is configured under twamp/server. However,
              in the event that the user re-configured
              server/max-count after this control connection is
              already in progress, this read-only value will show the
              actual max-count that is in use for this
              control connection.";
          }

          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.
              Salt MUST be generated pseudo-randomly (independently
              of anything else in the RFC) and is communicated to
              the Control-Client as part of the Server Greeting
              message.";
          }

          leaf server-iv {
            type binary {
              length 16;
            }
            description
              "The Server Initialization Vector
              (IV) is 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 session-sender;
      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
            "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;
            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 uint32;
                description
                  "Indicates the period to wait between the first bits
                  of TWAMP-Test (UDP) packet transmissions for
                  this test session";
              }
              leaf periodic-interval-units  {
                type time-units;
                description "Periodic interval time unit.";
                reference
                "RFC 3432: Network performance measurement
                           with periodic streams";
              }
            }
            case poisson {
              leaf lambda {
                type uint32;
                description
```

```
                "Indicates the average packet transmission rate.";
            }
            leaf lambda-units {
              type uint32;
              description
                "Indicates the units of lambda in
                reciprocal seconds.";
             reference
              "RFC 3432: Network performance measurement
                         with periodic streams";
            }
            leaf max-interval {
              type uint32;
              description
                "Indicates the maximum time between packet
                transmissions.";
```



```
            }
            leaf truncation-point-units {
              type time-units;
              description "Time units to truncate.";
            }
          }
        }

        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 session-reflector;
      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;
          }
          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.

              The default value of REFWAIT SHALL be 900 seconds, and
              this waiting time MAY be configurable. This timeout
              allows a Session-Reflector to free up resources in
              case of failure.";
```



```
        }

        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
              will be 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 test
              session.";
          }

          uses maintenance-statistics;
```



```
        }
      }
    }
  }
```

```
<CODE ENDS>
```

6. Data Model Examples

This section presents a simple but complete example of configuring all four entities in Figure 1, based on the YANG module specified in Section 5. The example is illustrative in nature, but aims to be self-contained, i.e. were it to be executed in a real TWAMP implementation it would lead to a correctly configured test session. 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

The following configuration example shows a Control-Client with client/admin-state enabled. In a real implementation following Figure 2 this would permit the initiation of TWAMP-Control connections and TWAMP-Test sessions.

```
<?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>
```

The following configuration 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>10.1.1.1</sender-ip>
                  <sender-udp-port>50000</sender-udp-port>
                  <reflector-ip>10.1.1.2</reflector-ip>
                  <reflector-udp-port>500001</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>4001</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:DB8:10:1:1::1</sender-ip>
                  <sender-udp-port>4000</sender-udp-port>
                  <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
                  <reflector-udp-port>5000</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>4001</sender-udp-port>
                  <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
                  <reflector-udp-port>5001</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

This configuration example shows a Server with server/admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Control connections and TWAMP-Test sessions.

```
<?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>
```

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

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>
              <periodic-interval-units>seconds</periodic-interval-units>
              <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>
              <lambda-units>1</lambda-units>
              <max-interval>2</max-interval>
              <truncation-point-units>seconds</truncation-point-units>
              <state>setup</state>
           </test-session>
        </session-sender>
```

```
    </twamp>
  </data>
```

6.4. Session-Reflector

The following example shows the two Session-Reflector TWAMP-Test sessions corresponding to the test sessions presented in Section 6.3.

```
<?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>10.1.1.1</sender-ip>
            <sender-udp-port>4000</sender-udp-port>
            <reflector-ip>10.1.1.2</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>50000</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>10.1.1.1</sender-ip>
               <sender-udp-port>4000</sender-udp-port>
               <reflector-ip>10.1.1.2</reflector-ip>
               <reflector-udp-port>5000</reflector-udp-port>
               <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>4001</sender-udp-port>
                <reflector-ip>192.68.0.2</reflector-ip>
                <reflector-udp-port>5001</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 and Robert Sherman for their thorough and constructive reviews, comments and text suggestions.

Haoxing Shen contributed to the definition of the YANG module in Section 5.

Jan Lindblad and Ladislav Lhokta did thorough reviews of the YANG module and the examples in Appendix A.

Kostas Pentikousis is partially supported by FP7 UNIFY (http://fp7-unify.eu), a research project partially funded by the European Community under the Seventh Framework Program (grant agreement no. 619609). The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

10. References

10.1. Normative References

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

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <http://www.rfc-editor.org/info/rfc4656>.

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

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

[RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features", RFC 6038, DOI 10.17487/RFC6038, October 2010, <http://www.rfc-editor.org/info/rfc6038>.

[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>.

10.2. Informative References

[I-D.ietf-ippm-metric-registry] Bagnulo, M., Claise, B., Eardley, P., Morton, A., and A. Akhter, "Registry for Performance Metrics", draft-ietf-ippm-metric-registry-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>10.1.1.1</sender-ip>
                  <sender-udp-port>4000</sender-udp-port>
                  <reflector-ip>10.1.1.2</reflector-ip>
```

```
                  <reflector-udp-port>5000</reflector-udp-port>
                  <padding-length>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>4001</sender-udp-port>
                  <reflector-ip>203.0.113.2</reflector-ip>
                  <reflector-udp-port>5001</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>4000</sender-udp-port>
                  <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
                  <reflector-udp-port>5000</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>4001</sender-udp-port>
                  <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
                  <reflector-udp-port>5001</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>
           <periodic-interval-units>
                seconds
           </periodic-interval-units>
           <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>
           <lambda-units>1</lambda-units>
           <max-interval>2</max-interval>
           <truncation-point-units>seconds</truncation-point-units>
           <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>10.1.1.1</sender-ip>
            <sender-udp-port>4000</sender-udp-port>
            <reflector-ip>10.1.1.2</reflector-ip>
            <reflector-udp-port>5000</reflector-udp-port>
            <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>4001</sender-udp-port>
            <reflector-ip>192.68.0.2</reflector-ip>
            <reflector-udp-port>5001</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>4000</sender-udp-port>
                <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
                <reflector-udp-port>5000</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>4001</sender-udp-port>
                <reflector-ip>2001:DB8:192:68::2</reflector-ip>
                <reflector-udp-port>5001</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

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com