IPPM WG
Internet-Draft
Intended status: Standards Track
Expires: October 16, 2018

R. Civil
Ciena Corporation
A. Morton
AT&T Labs
R. Rahman
Cisco Systems
M. Jethanandani

K. Pentikousis, Ed.
Travelping
April 14, 2018

Two-Way Active Measurement Protocol (TWAMP) Data Model
draft-ietf-ippm-twamp-yang-08

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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on October 16, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction 3
1.1. Motivation 3
1.2. Terminology 4
1.3. Document Organization 4
2. Scope, Model, and Applicability 4
3. Data Model Overview 5
3.1. Control-Client 6
3.2. Server 7
3.3. Session-Sender 7
3.4. Session-Reflector 8
4. Data Model Parameters 8
4.1. Control-Client 8
4.2. Server 11
4.3. Session-Sender 13
4.4. Session-Reflector 14
5. Data Model 16
5.1. YANG Tree Diagram 16
5.2. YANG Module 19
6. Data Model Examples 48
6.1. Control-Client 48
6.2. Server 50
6.3. Session-Sender 51
6.4. Session-Reflector 52
7. Security Considerations 55
8. IANA Considerations 56
9. Acknowledgements 56
10. Contributors 56
11. References 57
11.1. Normative References 57
11.2. Informative References 58
Appendix A. Detailed Data Model Examples 60
A.1. Control-Client 60
A.2. Server 62
A.3. Session-Sender 63
A.4. Session-Reflector 64
Appendix B. TWAMP Operational Commands 67
Authors' Addresses 67

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, implementors have no choice except to provide a proprietary mechanism. This document addresses this gap by formally specifying the TWAMP data model using YANG [RFC7950].

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 Unifying Carrier and Cloud Networks: Problem Statement and Challenges [I-D.unify-nfvrg-challenges]DevOps For Software-Defined Telecom Infrastructures [I-D.unify-nfvrg-devops], proprietary mechanisms for managing TWAMP measurements pose severe limitations with respect to programmability.

Two major trends call for standardizing TWAMP management aspects. First, we expect that in the coming years large-scale and multi-vendor TWAMP deployments will become the norm. From an operations perspective, using several vendor-specific TWAMP configuration mechanisms when one standard mechanism could provide an alternative is expensive and inefficient. Second, the increasingly software-defined and virtualized nature of network infrastructures, based on dynamic service chains [NSC] and programmable control and management planes Software-Defined Networking (SDN): Layers and Architecture Terminology [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 [RFC7950] data modeling language.

Note to RFC Editor:

Please replace the date 2018-04-16 in Section 5.2 of the draft with the date of publication of this draft as a RFC. Also, replace reference to RFC XXXX, and draft-ietf-port-twamp-test with the RFC numbers assigned to the drafts.

1.2. Terminology

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

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

```
    [Fig. 3]                                [Fig. 4]
+----------------+                        +--------+
| Control-Client |  <-- TWAMP-Control -->  | Server |
+----------------+                        +--------+
        ^                                      ^
        |                                      |
        V                                      V
+----------------+                    +-------------------+
| Session-Sender |  <-- TWAMP-Test --> | Session-Reflector |
+----------------+                    +-------------------+
    [Fig. 5]                                [Fig. 6]
```

Figure 1: Annotated TWAMP logical model

As per TWAMP [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 [RFC8040].

```
o-------------------o                      o-------------------o
|   Config client   |                      |   Config client   |
o-------------------o                      o-------------------o
         ||                                         ||
 NETCONF || RESTCONF                        NETCONF || RESTCONF
         ||                                         ||
o-------------------o                      o-------------------o
|   Config server   |                      |   Config server   |
|    [Fig. 3, 5]    |                      |    [Fig. 4, 6]    |
+-------------------+                      +-------------------+
|   Control-Client  | <-- TWAMP-Control --> |      Server       |
|                   |                      |                   |
|   Session-Sender  |  <-- TWAMP-Test -->  | Session-Reflector |
+-------------------+                      +-------------------+
```

Figure 2: Simplified TWAMP model and protocols

The data model defined in this document is orthogonal to the specific protocol used between the Config client and Config server to communicate the TWAMP configuration parameters.

Operational actions such as how TWAMP-Test sessions are started and stopped, how performance measurement results are retrieved, or how stored results are cleared, and so on, are not addressed by the configuration model defined in this document. As noted above, such operational actions are not part of the TWAMP specification TWAMP [RFC5357] and hence are out of scope of this document. See also Appendix B.

3. Data Model Overview

The TWAMP data model includes four categories of configuration items.

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

A second category includes attributes that can be configured on a per TWAMP-Control connection basis, such as the Server IP address.

A third category includes attributes related to per TWAMP-Test session attributes, for instance setting different values in the Differentiated Services Code Point (DSCP) field.

Finally, the data model includes attributes that relate to the operational state of the TWAMP implementation.

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

3.1. Control-Client

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

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

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

- o The IP address of the interface the Control-Client will use for connections.

- o The IP address of the remote TWAMP Server.

- o Authentication and encryption attributes such as KeyID, Token and the Client Initialization Vector (Client-IV); see also the last paragraph of Section 6 in OWAMP [RFC4656] and Randomness Requirements for Security [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:

- o The test session name that uniquely identifies a particular test session at the Control-Client and Session-Sender. Similarly to the control connections above, this unique test session name is

needed because at the time of creation of a TWAMP-Test session, for example, the source UDP port number is not known to uniquely identify the test session.

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

- o The IP address and UDP port number of the Session-Reflector on said path.

- o Information pertaining to the test packet stream, such as the test starting time, which performance metric is to be used Registry for Performance Metrics [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:

- o The test session name that MUST be identical with the corresponding test session name on the TWAMP Control-Client (Section 3.1).

- o The control connection name, which along with the test session name uniquely identify the TWAMP Session-Sender instance.

- o Information pertaining to the test packet stream, such as, for example, the number of test packets and the packet distribution to be employed; see also Network performence measurement with periodic streams [RFC3432].

3.4. Session-Reflector

Each TWAMP Session-Reflector has an administrative status field set at the device level to indicate whether the node is enabled to function as such.

Each Session-Reflector is associated with zero or more TWAMP-Test sessions. For each test session, the REFWAIT timeout parameter which determines whether to discontinue the session if no packets have been received (TWAMP [RFC5357], Section 4.2) can be configured.

Read-only access to other data model parameters, such as the Sender IP address is foreseen. Each test session can be uniquely identified by the 4-tuple mentioned in Section 3.2.

4. Data Model Parameters

This section defines the TWAMP data model using UML and introduces selected parameters associated with the four TWAMP logical entities. The complete TWAMP data model specification is provided in the YANG module presented in Section 5.2.

4.1. Control-Client

The client container (see Figure 3) holds items that are related to the configuration of the TWAMP Control-Client logical entity (recall Figure 1).

The client container includes an administrative configuration parameter (client/admin-state) that indicates whether the device is allowed to initiate TWAMP-Control connections.

```
+-------------+
| client      |
+-------------+                  1..* +-----------------------+
| admin-state |<>----------------------| mode-preference-chain |
|             |                        +-----------------------+
|             |  1..* +------------+   | priority              |
|             |<>-----| key-chain  |   | mode                  |
+-------------+       +------------+   +-----------------------+
       ^              | key-id     |
       V              | secret-key |
       |              +------------+
       | 0..*
+------------------------+
| ctrl-connection        |
+------------------------+
| name                   |
| client-ip              |
| server-ip              |
| server-tcp-port        |    0..* +----------------------+
| control-packet-dscp    |<>-------| test-session-request |
| key-id                 |         +----------------------+
| max-count              |         | name                 |
| client-tcp-port   {ro} |         | sender-ip            |
| server-start-time {ro} |         | sender-udp-port      |
| state             {ro} |         | reflector-ip         |
| selected-mode     {ro} |         | reflector-udp-port   |
| token             {ro} |         | timeout              |
| client-iv         {ro} |         | padding-length       |
+------------------------+         | test-packet-dscp     |
                                   | start-time           |
            +-------------+ 1      | repeat               |
            | pm-reg-list |------<>| repeat-interval      |
            +-------------+        | state           {ro} |
            | pm-index    |        | sid             {ro} |
            +-------------+        +----------------------+
```

Figure 3: TWAMP Control-Client UML class diagram

The client container holds a list (mode-preference-chain) which specifies the Mode values according to their preferred order of use by the operator of this Control-Client, including the authentication and encryption Modes. Specifically, mode-preference-chain lists the mode and its corresponding priority, expressed as a 16-bit unsigned integer, where zero is the highest priority and subsequent values are monotonically increasing.

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

Note that the list of preferred Modes may set bit position combinations when necessary, such as when referring to the extended TWAMP features in Mixed Security Mode for TWAMP [RFC5618], Individual Session Control Feature for TWAMP [RFC5938], TWAMP Reflect Octets and Symmetrical Size Features [RFC6038], and IKEv2-Derived Shared Secret Key for OWAMP and TWAMP [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 YANG [RFC7950]. The derived key length (dkLen in PKCS #5: Password-Based Cryptography Specification Version 2.1 [RFC8018]) MUST be 16 octets for the AES Session-key used for encryption and 32 octets for the HMAC-SHA1 Session-key used for authentication; see also Section 6.10 of OWAMP [RFC4656].

Each client container also holds a list of control connections, where each item in the list describes a TWAMP control connection initiated by this Control-Client. There SHALL be one ctrl-connection per TWAMP-Control (TCP) connection that is to be initiated from this device.

In turn, each ctrl-connection holds a list of test-session-request. test-session-request holds information associated with the Control-Client for this test session. This includes information associated with the Request-TW-Session/Accept-Session message exchange (see Section 3.5 of TWAMP [RFC5357]).

There SHALL be one instance of test-session-request for each TWAMP-Test session that is to be negotiated by this TWAMP-Control connection via a Request-TW-Session/Accept-Session exchange.

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

4.2. Server

The server container (see Figure 4) holds items that are related to the configuration of the TWAMP Server logical entity (recall Figure 1).

The server container includes an administrative configuration parameter (server/admin-state) that indicates whether the device is allowed to receive TWAMP-Control connections.

A device operating in the Server role cannot configure attributes on a per TWAMP-Control connection basis, as it has no foreknowledge of the incoming TWAMP-Control connections to be received. Consequently, any parameter that the Server might want to apply to an incoming control connection must be configured at the overall Server level and applied to all incoming TWAMP-Control connections.

```
+---------------------+
| server              |
+---------------------+
| admin-state         |   1..* +------------+
| server-tcp-port     |<>------| key-chain  |
| servwait            |        +------------+
| control-packet-dscp |        | key-id     |
| count               |        | secret-key |
| max-count           |        +------------+
| modes               |
|                     |   0..* +--------------------------+
|                     |<>------| ctrl-connection          |
+---------------------+        +--------------------------+
                               | client-ip           {ro} |
                               | client-tcp-port     {ro} |
                               | server-ip           {ro} |
                               | server-tcp-port     {ro} |
                               | state               {ro} |
                               | control-packet-dscp {ro} |
                               | selected-mode       {ro} |
                               | key-id              {ro} |
                               | count               {ro} |
                               | max-count           {ro} |
                               | salt                {ro} |
                               | server-iv           {ro} |
                               | challenge           {ro} |
                               +--------------------------+
```

Figure 4: TWAMP Server UML class diagram

Each server container holds a list named key-chain which relates KeyIDs with the respective secret keys. As mentioned in Section 4.1, both the Server and the Control-Client use the same mappings from KeyIDs to shared secrets. The Server, being prepared to conduct sessions with more than one Control-Client, uses KeyIDs to choose the appropriate secret-key; a Control-Client would typically have different secret keys for different Servers. key-id tells the Server which shared-secret the Control-Client wishes to use for authentication or encryption.

Each incoming control connection active on the Server is represented by a ctrl-connection. There SHALL be one ctrl-connection per incoming TWAMP-Control (TCP) connection that is received and active on the Server. Each ctrl-connection can be uniquely identified by the 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port}. All items in the ctrl-connection list are read-only.

4.3. Session-Sender

The session-sender container, illustrated in Figure 5, holds items that are related to the configuration of the TWAMP Session-Sender logical entity.

The session-sender container includes an administrative parameter (session-sender/admin-state) that controls whether the device is allowed to initiate TWAMP-Test sessions.

```
+----------------+
| session-sender |
+----------------+  0..* +---------------------------+
| admin-state    |<>-----| test-session              |
+----------------+       +---------------------------+
                         | name                      |
                         | ctrl-connection-name {ro} |
                         | fill-mode                 |
                         | number-of-packets         |
                         | state {ro}                |
                         | sent-packets         {ro} |
                         | rcv-packets          {ro} |
                         | last-sent-seq        {ro} |
                         | last-rcv-seq         {ro} |
                         +---------------------------+
                                     ^
                                     V
                                     | 1
                          +---------------------+
                          | packet-distribution |
                          +---------------------+
                          | periodic /  poisson |
                          +---------------------+
                              |           |
                     +-------------------+  |
                     | periodic-interval |  |
                     +-------------------+  |
                                            |
                                  +--------------+
                                  | lambda       |
                                  | max-interval |
                                  +--------------+
```

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.

```
+----=--------------+
| session-reflector |
+-------------------+
| admin-state       |
| refwait           |
+-------------------+
         ^
         V
         |
         | 0..*
+----------------------------------------+
| test-session                           |
+----------------------------------------+
| sid                                {ro} |
| sender-ip                          {ro} |
| sender-udp-port                    {ro} |
| reflector-ip                       {ro} |
| reflector-udp-port                 {ro} |
| parent-connection-client-ip        {ro} |
| parent-connection-client-tcp-port  {ro} |
| parent-connection-server-ip        {ro} |
| parent-connection-server-tcp-port  {ro} |
| test-packet-dscp                   {ro} |
| sent-packets                       {ro} |
| rcv-packets                        {ro} |
| last-sent-seq                      {ro} |
| last-rcv-seq                       {ro} |
+----------------------------------------+
```

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 TWAMP Reflect Octets and Symmetrical Size Features [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, and then pick out specific test-session instances of interest to the user. This could be problematic if a large number of test sessions are currently active on that device.

Each Session-Reflector TWAMP-Test session contains the following 4-tuple: {parent-connection-client-ip, parent-connection-client-tcp-port, parent-connection-server-ip, parent-connection-server-tcp-port}. This 4-tuple MUST correspond to the equivalent 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port} in server/ctrl-connection. This 4-tuple allows the user to trace back from the TWAMP-Test session to the (parent) TWAMP-Control connection that negotiated this test session.

5. Data Model

This section formally specifies the TWAMP data model using YANG.

5.1. YANG Tree Diagram

This section presents a simplified graphical representation of the TWAMP data model using a YANG tree diagram. Readers should keep in mind that the limit of 72 characters per line forces us to introduce artificial line breaks in some tree diagram nodes. Tree diagrams used in this document follow the notation defined in YANG Tree Diagrams [RFC8340].

```
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?              uint8
         |     +--ro client-tcp-port?        inet:port-number
         |     +--ro server-start-time?      uint64
         |     +--ro repeat-count?           uint64
         |     +--ro state?
         |     |       control-client-connection-state
         |     +--ro selected-mode?          twamp-modes
         |     +--ro token?                  binary
         |     +--ro client-iv?              binary
         |     +--rw test-session-request* [name]
         |        +--rw name                  string
         |        +--rw sender-ip?            inet:ip-address
         |        +--rw sender-udp-port?      union
         |        +--rw reflector-ip          inet:ip-address
         |        +--rw reflector-udp-port?   inet:port-number
         |        +--rw timeout?              uint64
         |        +--rw padding-length?       uint32
         |        +--rw test-packet-dscp?     inet:dscp
         |        +--rw start-time?           uint64
         |        +--rw repeat?               union
         |        +--rw repeat-interval?      uint32
         |        +--rw pm-reg-list* [pm-index]
         |        |  +--rw pm-index    uint16
         |        +--ro state?                test-session-state
         |        +--ro sid?                  string
        +--rw server {server}?
         |  +--rw admin-state?           boolean
         |  +--rw server-tcp-port?       inet:port-number
         |  +--rw servwait?              uint32
         |  +--rw control-packet-dscp?   inet:dscp
         |  +--rw count?                 uint8
         |  +--rw max-count?             uint8
         |  +--rw modes?                 twamp-modes
         |  +--rw key-chain* [key-id]
         |  |  +--rw key-id        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?                 uint8
         |     +--ro max-count?             uint8
```

```
        |     +--ro salt?                  binary
        |     +--ro server-iv?             binary
        |     +--ro challenge?             binary
        +--rw session-sender {session-sender}?
        |  +--rw admin-state?    boolean
        |  +--rw test-session* [name]
        |     +--rw name                    string
        |     +--ro ctrl-connection-name?   string
        |     +--rw fill-mode?              padding-fill-mode
        |     +--rw number-of-packets       uint32
        |     +--rw (packet-distribution)?
        |     |  +--:(periodic)
        |     |  |  +--rw periodic-interval       decimal64
        |     |  +--:(poisson)
        |     |     +--rw lambda                  decimal64
        |     |     +--rw max-interval?           decimal64
        |     +--ro state?                  sender-session-state
        |     +--ro sent-packets?           uint32
        |     +--ro rcv-packets?            uint32
        |     +--ro last-sent-seq?          uint32
        |     +--ro last-rcv-seq?           uint32
        +--rw session-reflector {session-reflector}?
           +--rw admin-state?    boolean
           +--rw refwait?        uint32
           +--ro test-session*
                   [sender-ip sender-udp-port reflector-ip reflector-udp
 -port]
             +--ro sid?                                string
             +--ro sender-ip                           inet:ip-address
             +--ro sender-udp-port
             |       dynamic-port-number
             +--ro reflector-ip                        inet:ip-address
             +--ro reflector-udp-port                  inet:port-numbe
 r
             +--ro parent-connection-client-ip?        inet:ip-address
             +--ro parent-connection-client-tcp-port?  inet:port-numbe
 r
             +--ro parent-connection-server-ip?        inet:ip-address
             +--ro parent-connection-server-tcp-port?  inet:port-numbe
 r
             +--ro test-packet-dscp?                   inet:dscp
             +--ro sent-packets?                       uint32
             +--ro rcv-packets?                        uint32
             +--ro last-sent-seq?                      uint32
             +--ro last-rcv-seq?                       uint32
```

Figure 7: YANG Tree Diagram.

5.2. YANG Module

This section presents the YANG module for the TWAMP data model defined in this document. The module imports definitions from NTPv3 Specification [RFC1305], Randomness Requirements for Security [RFC4086], OWAMP [RFC4656], TWAMP [RFC5357], More Features for TWAMP [RFC5618], Individual Session Control Feature [RFC5938], TWAMP Reflect Octets and Symmetrical Size Features [RFC6038], Common YANG Data Types [RFC6991], Advances Stream and Sampling Framework [RFC7312], IKEv2-Derived Shared Secret Key for OWAMP and TWAMP [RFC7717], and OWAMP and TWAMP Well-Known Port Assignments [I-D.ietf-ippm-port-twamp-test].

```
<CODE BEGINS> file "ietf-twamp@2018-04-16.yang"

module ietf-twamp {
  yang-version 1.1;
  namespace urn:ietf:params:xml:ns:yang:ietf-twamp;
  prefix ietf-twamp;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Types.";
  }

  organization
    "IETF IPPM (IP Performance Metrics) Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/ippm/
     WG List: ippm@ietf.org

     Editor: Ruth Civil
             gcivil@ciena.com
     Editor: Al Morton
             acmorton@att.com
     Editor: Reshad Rehman
             rrahman@cisco.com
     Editor: Mahesh Jethanandani
             mjethanandani@gmail.com
     Editor: Kostas Pentikousis
             k.pentikousis@travelping.com";


  description
    "This YANG module specifies a vendor-independent data
     model for the Two-Way Active Measurement Protocol (TWAMP).
```

```
        The data model covers four TWAMP logical entities, namely,
        Control-Client, Server, Session-Sender, and Session-Reflector,
        as illustrated in the annotated TWAMP logical model (Fig. 1
        of RFC XXXX).

        This YANG module uses features to indicate which of the four
        logical entities are supported by a TWAMP implementation.

        Copyright (c) 2018 IETF Trust and the persons identified as
        the document authors.  All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD
        License set forth in Section 4.c of the IETF Trust's Legal
        Provisions Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

      revision 2018-04-16 {
        description
          "Initial Revision.

          Covers RFC 5357, RFC 5618, RFC 5938, RFC 6038, RFC 7717, and
          draft-ietf-ippm-metric-registry";

        reference
          "RFC XXXX: TWAMP YANG Data Model.";
      }


      /*
       * Typedefs
       */

      typedef twamp-modes {
        type bits {
          bit unauthenticated {
            position 0;
            description
              "Unauthenticated mode, in which no encryption or
              authentication is applied in TWAMP-Control and
              TWAMP-Test. KeyID, Token, and Client-IV are not used in
              the Set-Up-Response message. See Section 3.1 of
              RFC 4656.";
            reference
              "RFC 4656: A One-way Active Measurement Protocol
```

```
            (OWAMP)";
        }
        bit authenticated {
          position 1;
          description
            "Authenticated mode, in which the Control-Client and
             Server possess a shared secret thus prohibiting
             'theft of service'. As per Section 6 of RFC 4656,
             in 'authenticated mode, the timestamp is in the clear
             and is not protected cryptographically in any way,
             while the rest of the message has the same protection
             as in encrypted mode. This mode allows one to trade off
             cryptographic protection against accuracy of
             timestamps.'";
          reference
            "RFC 4656: A One-way Active Measurement Protocol
             (OWAMP)";
        }
        bit encrypted {
          position 2;
          description
            "Encrypted mode 'makes it impossible to alter
             timestamps undetectably.' See also Section 4 of RFC 7717
             and Section 6 of RFC 4656.";
          reference
            "RFC 4656: A One-way Active Measurement Protocol
             (OWAMP)";
        }
        bit unauth-test-encrpyt-control {
          position 3;
          description
            "When using the Mixed Security Mode, the TWAMP-Test
             protocol follows the Unauthenticated mode and the
             TWAMP-Control protocol the Encrypted mode.";
          reference
            "RFC 5618: Mixed Security Mode for the Two-Way Active
             Measurement Protocol (TWAMP)";
        }
        bit individual-session-control {
          position 4;
          description
            "This mode enables individual test sessions using
             Session Identifiers.";
          reference
            "RFC 5938: Individual Session Control Feature
             for the Two-Way Active Measurement Protocol (TWAMP)";
        }
        bit reflect-octets {
```

```
          position 5;
          description
            "This mode indicates the reflect octets capability.";
          reference
            "RFC 6038: Two-Way Active Measurement Protocol (TWAMP)
             Reflect Octets and Symmetrical Size Features";
        }
        bit symmetrical-size {
          position 6;
          description
            "This mode indicates support for the symmetrical size
             sender test packet format.";
          reference
            "RFC 6038: Two-Way Active Measurement Protocol (TWAMP)
             Reflect Octets and Symmetrical Size Features";
        }
        bit IKEv2Derived {
          position 7;
          description
            "In this mode the the shared key is derived
             from an IKEv2 security association (SA).";
          reference
            "RFC 7717: IKEv2-Derived Shared Secret Key for
             the One-Way Active Measurement Protocol (OWAMP)
             and Two-Way Active Measurement Protocol (TWAMP)";
        }
      }
      description
        "Specifies the configurable TWAMP-Modes supported during a
         TWAMP-Control Connection setup between a Control-Client
         and a Server. Section 7 of RFC 7717 summarizes the
         TWAMP-Modes registry and points to their formal
         specification.";
    }

    typedef control-client-connection-state {
      type enumeration {
        enum active {
          description
            "Indicates an active TWAMP-Control connection to
             Server.";
        }
        enum idle {
          description
            "Indicates an idle TWAMP-Control connection to Server.";
        }
      }
      description
```

```
        "Indicates the Control-Client TWAMP-Control connection
         state.";
    }

    typedef test-session-state {
      type enumeration {
        enum accepted {
          value 0;
          description
            "Indicates an accepted TWAMP-Test session request.";
        }
        enum failed {
          value 1;
          description
            "Indicates a TWAMP-Test session failure due to
             some unspecified reason (catch-all).";
        }
        enum internal-error {
          value 2;
          description
            "Indicates a TWAMP-Test session failure due to
             an internal error.";
        }
        enum not-supported {
          value 3;
          description
            "Indicates a TWAMP-Test session failure because
             some aspect of the TWAMP-Test session request
             is not supported.";
        }
        enum permanent-resource-limit {
          value 4;
          description
            "Indicates a TWAMP-Test session failure due to
             permanent resource limitations.";
        }
        enum temp-resource-limit {
          value 5;
          description
            "Indicates a TWAMP-Test session failure due to
             temporary resource limitations.";
        }
      }
      description
        "Indicates the Control-Client TWAMP-Test session state.";
    }

    typedef server-ctrl-connection-state {
```

```
        type enumeration {
          enum active {
            description
              "Indicates an active TWAMP-Control connection
               to the Control-Client.";
          }
          enum servwait {
            description
              "Indicates that the TWAMP-Control connection to the
               Control-Client is in SERVWAIT as per the definition of
               Section 3.1 of RFC 5357.";
          }
        }
        description
          "Indicates the Server TWAMP-Control connection state.";
      }

      typedef sender-session-state {
        type enumeration {
          enum active {
            description
              "Indicates that the TWAMP-Test session is active.";
          }
          enum failure {
            description
              "Indicates that the TWAMP-Test session has failed.";
          }
        }
        description
          "Indicates the Session-Sender TWAMP-Test session state.";
      }

      typedef padding-fill-mode {
        type enumeration {
          enum zero {
            description
              "TWAMP-Test packets are padded with all zeros.";
          }
          enum random {
            description
              "TWAMP-Test packets are padded with pseudo-random
              numbers.";
          }
        }
        description
          "Indicates what type of packet padding is used in the
           TWAMP-Test packets.";
      }
```

```
typedef dynamic-port-number {
  type inet:port-number {
    range 49152..65535;
  }
  description "Dynamic range for port numbers.";
}

/*
 * Features
 */

feature control-client {
  description
    "Indicates that the device supports configuration of the
     TWAMP Control-Client logical entity.";
}

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

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

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


/*
 * Reusable node groups
 */

grouping key-management {
  list key-chain {
    key key-id;
    leaf key-id {
      type string {
        length 1..80;
      }
      description
```

```
              "KeyID used for a TWAMP-Control connection. As per
               Section 3.1 of RFC 4656, KeyID is 'a UTF-8 string, up to
               80 octets in length' and is used to select which 'shared
               shared secret the [Control-Client] wishes to use to
               authenticate or encrypt'.";
            }
            leaf secret-key {
              type string;
              description
                "The secret key corresponding to the KeyID for this
                 TWAMP-Control connection.";
            }
            description
              "Relates KeyIDs with their respective secret keys
               in a TWAMP-Control connection.";
        }
        description
            "Used by the Control-Client and Server for TWAMP-Control
             key management.";
      }

      grouping maintenance-statistics {
        leaf sent-packets {
          type uint32;
          config false;
          description
            "Indicates the number of packets sent.";
        }

        leaf rcv-packets {
          type uint32;
          config false;
          description
            "Indicates the number of packets received.";
        }

        leaf last-sent-seq {
          type uint32;
          config false;
          description
            "Indicates the last sent sequence number.";
        }

        leaf last-rcv-seq {
          type uint32;
          config false;
          description
            "Indicates the last received sequence number.";
```

```
      }
      description
        "Used for TWAMP-Test maintenance statistics.";
    }

    grouping count {
      leaf count {
        type uint8 {
          range "10..31";
        }
        default 10;
        description
          "Parameter communicated to the Control-Client as part of
           the Server Greeting message and used for deriving a key
           from a shared secret as per Section 3.1 of  RFC 4656:
           MUST be a power of 2 and at least 1024. It is configured
           by providing said power. For example, configuring 15 here
           means count 2^15 = 32768. The default is 10,
           meaning 2^10 = 1024.";
        }
     description
       "Reusable data structure for count, which is used both in the
        Server and the Control-Client.";
    }

    grouping max-count-exponent {
      leaf max-count {
        type uint8 {
          range 10..31;
        }
        default 15;
        description
          "This parameter limits the maximum Count value, which MUST
           be a power of 2 and at least 1024 as per RFC 5357. It is
           configured by providing said power. For example,
           configuring 10 here means max count 2^10 = 1024.
           The default is 15, meaning 2^15 = 32768.

           A TWAMP Server uses this configured value in the
           Server-Greeting message sent to the Control-Client.

           A TWAMP Control-Client uses this configured value to
           prevent denial-of-service (DOS) attacks by closing the
           control connection to the Server if it 'receives a
           Server-Greeting message with Count greater that its
           maximum configured value', as per Section 6 of RFC 5357.

           Further, note that according to Section 6 of RFC 5357:
```

```
            'If an attacking system sets the maximum value in
            Count (2**32), then the system under attack would stall
            for a significant period of time while it attempts to
            generate keys.

            TWAMP-compliant systems SHOULD have a configuration
            control to limit the maximum count value. The default
            max-count value SHOULD be 32768.'

            RFC 5357 does not qualify 'significant period' in terms of
            time, but it is clear that this depends on the processing
            capacity available and operators need to pay attention to
            this security consideration.";
        }
      description
        "Reusable data structure for max-count which is used both at
         the Control-Client and the Server containers.";
    }


    /*
     * Configuration data nodes
     */

    container twamp {
      description
        "TWAMP logical entity configuration grouping of four models
        which correspond to the four TWAMP logical entities
        Control-Client, Server, Session-Sender, and Session-Reflector
        as illustrated in Fig. 1 of RFC XXXX.";

      container client {
        if-feature control-client;
        description
          "Configuration of the TWAMP Control-Client logical
           entity.";

        leaf admin-state {
          type boolean;
          default true;
          description
            "Indicates whether the device is allowed to operate as a
             TWAMP Control-Client.";
        }

        list mode-preference-chain {
          key priority;
          unique mode;
```

```
            leaf priority {
              type uint16;
              description
                "Indicates the Control-Client Mode preference priority
                 expressed as a 16-bit unsigned integer, where zero is
                 the highest priority and subsequent values
                 monotonically increasing.";
            }
            leaf mode {
              type twamp-modes;
              description
                "The supported TWAMP Mode matching the corresponding
                 priority.";

            }
            description
              "Indicates the Control-Client preferred order of use of
               the supported TWAMP Modes.

               Depending on the Modes available in the TWAMP Server
               Greeting message (see Fig. 2 of RFC 7717), the
               this Control-Client MUST choose the highest priority
               Mode from the configured mode-preference-chain list.";
          }

          uses key-management;

          list ctrl-connection {
            key name;
            description
              "List of TWAMP Control-Client control connections.
               Each item in the list describes a control connection
               that will be initiated by this Control-Client";

            leaf name {
              type string;
              description
                "A unique name used as a key to identify this
                 individual TWAMP-Control connection on the
                 Control-Client device.";
            }
            leaf client-ip {
              type inet:ip-address;
              description
                "The IP address of the local Control-Client device,
                 to be placed in the source IP address field of the
                 IP header in TWAMP-Control (TCP) packets belonging
                 to this control connection. If not configured, the
```

```
                 device SHALL choose its own source IP address.";
            }
            leaf server-ip {
              type inet:ip-address;
              mandatory true;
              description
                "The IP address of the remote Server device, which the
                 TWAMP-Control connection will be initiated to.";
            }

            leaf server-tcp-port {
              type inet:port-number;
              default 862;
              description
                "This parameter defines the TCP port number that is
                 to be used by this outgoing TWAMP-Control connection.
                 Typically, this is the well-known TWAMP-Control
                 port number (862) as per RFC 5357 However, there are
                 known realizations of TWAMP in the field that were
                 implemented before this well-known port number was
                 allocated. These early implementations allowed the
                 port number to be configured. This parameter is
                 therefore provided for backward compatibility
                 reasons.";
            }

            leaf control-packet-dscp {
              type inet:dscp;
              default 0;
              description
                "The DSCP value to be placed in the IP header of
                 TWAMP-Control (TCP) packets generated by this
                 Control-Client.";
            }

            leaf key-id {
              type string {
                length 1..80;
              }
              description
               "Indicates the KeyID value selected for this
                TWAMP-Control connection.";
            }

            uses max-count-exponent;

            leaf client-tcp-port {
              type inet:port-number;
```

```
            config false;
            description
              "Indicates the source TCP port number used in the
               TWAMP-Control packets belonging to this control
               connection.";
          }

          leaf server-start-time {
            type uint64;
            config false;
            description
              "Indicates the Start-Time advertized by the Server in
               the Server-Start message (RFC 4656, Section 3.1),
               representing the time when the current
               instantiation of the Server started operating.
               The timestamp format follows RFC 1305
               according to Section 4.1.2 of RFC 4656.";
            reference
              "RFC 4656: OWAMP, Section 3.1 and 4.1.2,
               RFC 1305: NTPv3 Specification.";
          }

          leaf repeat-count {
            type uint64;
            config false;
            description
              "Indicates how many times the test session has been
               repeated. When a test is running, this value will be
               greater than 0. If the repeat parameter is non-zero,
               this value is smaller than or equal to the repeat
               parameter.";
          }
          leaf state {
            type control-client-connection-state;
            config false;
            description
              "Indicates the current state of the TWAMP-Control
               connection state.";
          }

          leaf selected-mode {
            type twamp-modes;
            config false;
            description
              "The TWAMP Mode that the Control-Client has chosen for
               this control connection as set in the Mode field of
               the Set-Up-Response message";
            reference
```

```
                "RFC 4656, Section 3.1.";
            }

            leaf token {
                type binary {
                length 64;
              }
              config false;
              description
                "This parameter holds the 64 octets containing the
                 concatenation of a 16-octet Challenge, a 16-octet AES
                 Session-key used for encryption, and a 32-octet
                 HMAC-SHA1 Session-key used for authentication; see
                 also the last paragraph of Section 6 in RFC 4656.

                 If the Mode defined in RFC 7717 is selected
                 (selected-mode), Token is limited to 16 octets.";
              reference
                "RFC 4086: Randomness Requirements for Security

                 RFC 7717: IKEv2-Derived Shared Secret Key for the
                 One-Way Active Measurement Protocol (OWAMP) and
                 Two-Way Active Measurement Protocol (TWAMP)";
            }

            leaf client-iv {
              type binary {
                length 16;
              }
              config false;
              description
                "Indicates the Control-Client Initialization Vector
                 (Client-IV), that is generated randomly by the
                 Control-Client. As per RFC 4656:

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

                  If the Mode defined in RFC 7717 is selected
                  (selected-mode), Client-IV is limited to 12 octets.";
              reference
                "RFC 4656: A One-way Active Measurement Protocol
                 (OWAMP).
```

```
                 RFC 7717: IKEv2-Derived Shared Secret Key for the
                 One-Way Active Measurement Protocol (OWAMP) and
                 Two-Way Active Measurement Protocol (TWAMP)";
           }

           list test-session-request {
             key name;
             description
               "Information associated with the Control-Client
                for this test session";

             leaf name {
               type string;
               description
                 "A unique name to be used for identification of
                  this TWAMP-Test session on the Control-Client.";
             }

             leaf sender-ip {
               type inet:ip-address;
               description
                 "The IP address of the Session-Sender device,
                  which is to be placed in the source IP address
                  field of the IP header in TWAMP-Test (UDP) packets
                  belonging to this test session. This value will be
                  used to populate the sender address field of the
                  Request-TW-Session message.

                  If not configured, the device SHALL choose its own
                  source IP address.";
             }

             leaf sender-udp-port {
               type union {
                 type dynamic-port-number;
                 type enumeration {
                   enum autoallocate {
                     description
                       "Indicates that the Contol-Client will
                        auto-allocate the TWAMP-Test (UDP) port number
                        from the dynamic port range.";
                   }
                 }
               }
               default autoallocate;
               description
                 "The UDP port number that is to be used by
                  the Session-Sender for this TWAMP-Test session.
```

```
                 The number is restricted to the dynamic port range.

                 By default the Control-Client SHALL auto-allocate a
                 UDP port number for this TWAMP-Test session.

                 The configured (or auto-allocated) value is
                     advertized in the Sender Port field of the
                 Request-TW-session message (see Section 3.5 of
                 RFC 5357). Note that in the scenario where a device
                 auto-allocates a UDP port number for a session, and
                 the repeat parameter for that session indicates that
                 it should be repeated, the device is free to
                 auto-allocate a different UDP port number when it
                 negotiates the next (repeated) iteration of this
                 session.";
             }

             leaf reflector-ip {
               type inet:ip-address;
               mandatory true;
               description
                 "The IP address belonging to the remote
                  Session-Reflector device to which the TWAMP-Test
                  session will be initiated. This value will be
                  used to populate the receiver address field of
                  the Request-TW-Session message.";
             }

             leaf reflector-udp-port {
               type inet:port-number {
                 range "862 | 49152..65535";
               }
               description
                 "This parameter defines the UDP port number that
                  will be used by the Session-Reflector for
                  this TWAMP-Test session. The default number is
                  within the dynamic port range and is to be placed
                  in the Receiver Port field of the Request-TW-Session
                  message. The well-known port (862) MAY be
                  used.";
               reference
                 "draft-ietf-ippm-port-twamp-test: OWAMP and TWAMP
                  Well-Known Port Assignments.";
             }

             leaf timeout {
               type uint64;
               units seconds;
```

```
                default 2;
                description
                  "The length of time (in seconds) that the
                   Session-Reflector should continue to respond to
                   packets belonging to this TWAMP-Test session after
                   a Stop-Sessions TWAMP-Control message has been
                   received.

                   This value will be placed in the Timeout field of
                   the Request-TW-Session message.";
                reference
                  "RFC 5357: TWAMP, Section 3.8";
              }

              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.";
                reference
                  "RFC 4656, Section 3.5.";
              }

              leaf test-packet-dscp {
                type inet:dscp;
                default 0;
                description
                  "The DSCP value to be placed in the IP header
                   of TWAMP-Test packets generated by the
                   Session-Sender, and in the UDP header of the
                   TWAMP-Test response packets generated by the
                   Session-Reflector for this test session.

                   This value will be placed in the Type-P Descriptor
                   field of the Request-TW-Session message";
                reference
                  "RFC 5357.";
              }

              leaf start-time {
                type uint64;
                default 0;
```

```
                description
                  "Time when the session is to be started
                   (but not before the TWAMP Start-Sessions command
                   is issued; see Section 3.4 of RFC 5357).

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

                   The timestamp format follows RFC 1305 as per
                   Section 3.5 of RFC 4656.

                   The default value of 0 indicates that the session
                   will be started as soon as the Start-Sessions
                   message is received.";
              }

              leaf repeat {
                type union {
                  type uint32 {
                    range 0..4294967294;
                  }
                  type enumeration {
                    enum forever {
                      description
                        "Indicates that the test session SHALL be
                         repeated *forever* using the information in
                         repeat-interval parameter, and SHALL NOT
                         decrement the value.";
                    }
                  }
                }
                default 0;
                description
                  "This value determines if the TWAMP-Test session must
                   be repeated. When a test session has completed, the
                   repeat parameter is checked.

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

                   If the repeat value is 1 through 4,294,967,294
                   then the test session SHALL be repeated using the
                   information in repeat-interval parameter, and the
                   parent TWAMP-Control connection for this test
                   session is restarted to negotiate a new instance
                   of this TWAMP-Test session.";
              }
```

```
              leaf repeat-interval  {
                when "../repeat!='0'" {
                  description
                    "This parameter determines the timing of repeated
                     TWAMP-Test sessions when repeat is more than 0.

                     When the value of repeat-interval is 0, the
                     negotiation of a new test session SHALL begin
                     immediately after the previous test session
                     completes. Otherwise, the Control-Client will
                     wait for the number of seconds specified in the
                     repeat-interval parameter before negotiating the
                     new instance of this TWAMP-Test session.";
                }
                type uint32;
                units seconds;
                default 0;
                description
                  "Repeat interval (in seconds).";
              }

              list pm-reg-list {
                key pm-index;
                leaf pm-index {
                  type uint16;
                  description
                    "Numerical index value of a Registered Metric
                     in the Performance Metric Registry
                     (see ietf-ippm-metric-registry). Output statistics
                     are specified in the corresponding Registry
                     entry.";
                }
                description
                  "A list of one or more Performance Metric Registry
                   Index values, which communicate packet stream
                   characteristics along with one or more metrics
                   to be measured.

                   All members of the pm-reg-list MUST have the same
                   stream characteristics, such that they combine
                   to specify all metrics that shall be measured on
                   a single stream.";
                reference
                  "ietf-ippm-metric-registry: Registry for
                   Performance Metrics";
              }

              leaf state {
```

```
              type test-session-state;
              config false;
              description
                "Indicates the TWAMP-Test session state, accepted or
                 indication of an error.";
              reference
                "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";
              reference
                "Section 4.3 of RFC 6038.";
            }
          }
        }
      }

      container server {
        if-feature server;
        description
          "Configuration of the TWAMP Server logical entity.";

        leaf admin-state {
          type boolean;
          default true;
          description
            "Indicates whether the device is allowed to operate
             as a TWAMP Server.";
        }

        leaf server-tcp-port {
          type inet:port-number;
          default 862;
          description
            "This parameter defines the well known TCP port number
             that is used by TWAMP-Control. The Server will listen
             on this port number for incoming TWAMP-Control
             connections. Although this is defined as a fixed value
             (862) in RFC 5357, there are several realizations of
             TWAMP in the field that were implemented before this
             well-known port number was allocated. These early
             implementations allowed the port number to be
             configured. This parameter is therefore provided for
```

```
             backward compatibility reasons.";
         }

         leaf servwait {
           type uint32 {
             range 1..604800;
           }
           units seconds;
           default 900;
           description
             "TWAMP-Control (TCP) session timeout, in seconds.
              According to Section 3.1 of RFC 5357,

              Server MAY discontinue any established control
              connection when no packet associated with that
              connection has been received within SERVWAIT seconds.";
         }

         leaf control-packet-dscp {
           type inet:dscp;
           description
             "The DSCP value to be placed in the IP header of
              TWAMP-Control (TCP) packets generated by the Server.

              Section 3.1 of  RFC 5357 specifies that the server
              SHOULD use the DSCP value from the Control-Clients
              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-Clients TCP SYN.";
            reference
              "Section 3.1 of RFC 5357.";
         }

         uses count;

         uses max-count-exponent;

         leaf modes {
           type twamp-modes;
           description
             "The bit mask of TWAMP Modes this Server instance
```

```
              is willing to support; see IANA TWAMP Modes Registry.";
          }

          uses key-management;

          list ctrl-connection {
            key "client-ip client-tcp-port server-ip server-tcp-port";
            config false;
            description
              "List of all incoming TWAMP-Control (TCP) connections.";

            leaf client-ip {
              type inet:ip-address;
              description
                "The IP address on the remote Control-Client device,
                 which is the source IP address used in the
                 TWAMP-Control (TCP) packets belonging to this control
                 connection.";
            }

            leaf client-tcp-port {
              type inet:port-number;
              description
                "The source TCP port number used in the TWAMP-Control
                 (TCP) packets belonging to this control connection.";
            }

            leaf server-ip {
              type inet:ip-address;
              description
                "The IP address of the local Server device, which is
                 the destination IP address used in the
                 TWAMP-Control (TCP) packets belonging to this control
                 connection.";
            }

            leaf server-tcp-port {
              type inet:port-number;
              description
                "The destination TCP port number used in the
                 TWAMP-Control (TCP) packets belonging to this
                 control connection. This will usually be the
                 same value as the server-tcp-port configured
                 under twamp/server. However, in the event that
                 the user re-configured server/server-tcp-port
                 after this control connection was initiated, this
                 value will indicate the server-tcp-port that is
                 actually in use for this control connection.";
```

```
            }

            leaf state {
              type server-ctrl-connection-state;
              description
                "Indicates the Server TWAMP-Control connection state.";
            }

            leaf control-packet-dscp {
              type inet:dscp;
              description
                "The DSCP value used in the IP header of the
                 TWAMP-Control (TCP) packets sent by the Server
                 for this control connection. This will usually
                 be the same value as is configured in the
                 control-packet-dscp parameter under the twamp/server
                 container.  However, in the event that the user
                 re-configures server/dscp after this control
                 connection is already in progress, this read-only
                 value will show the actual dscp value in use by this
                 TWAMP-Control connection.";
            }

            leaf selected-mode {
              type twamp-modes;
              description
                "The Mode that was chosen for this TWAMP-Control
                 connection as set in the Mode field of the
                 Set-Up-Response message.";
            }

            leaf key-id {
              type string {
                length 1..80;
              }
              description
                "The KeyID value that is in use by this TWAMP-Control
                 connection as selected by Control-Client.";
            }

            uses count {
              description
                "The count value that is in use by this TWAMP-Control
                 connection. This will usually be the same value
                 as is configured under twamp/server. However, in the
                 event that the user re-configured server/count
                 after this control connection is already in progress,
                 this read-only value will show the actual count that
```

```
                 is in use for this TWAMP-Control connection.";
           }

           uses max-count-exponent {
             description
               "This read-only value indicates the actual max-count in
                use for this control connection. Usually this would be
                the same value as configured under twamp/server.";
           }

           leaf salt {
             type binary {
               length 16;
             }
             description
               "A parameter used in deriving a key from a
                shared secret as described in Section 3.1 of RFC 4656.
                It is communicated to the Control-Client as part of
                the Server Greeting message.";
           }

           leaf server-iv {
             type binary {
               length 16;
             }
             description
               "The Server Initialization Vector
                (IV) generated randomly by the Server.";
           }

           leaf challenge {
             type binary {
               length 16;
             }
             description
               "A random sequence of octets generated by the Server.
                As described in client/token, Challenge is used
                by the Control-Client to prove possession of a
                shared secret.";
           }
         }
       }

       container session-sender {
         if-feature session-sender;
         description
           "Configuration of the TWAMP Session-Sender logical entity";
         leaf admin-state {
```

```
            type boolean;
            default true;
            description
              "Indicates whether the device is allowed to operate
               as a TWAMP Session-Sender.";
          }

          list test-session{
            key name;
            description
              "List of TWAMP Session-Sender test sessions.";

            leaf name {
              type string;
              description
                "A unique name for this TWAMP-Test session to be used
                 for identifying this test session by the
                 Session-Sender logical entity.";
            }

            leaf ctrl-connection-name {
              type string;
              config false;
              description
                "The name of the parent TWAMP-Control connection that
                 is responsible for negotiating this TWAMP-Test
                 session.";
            }

            leaf fill-mode {
              type padding-fill-mode;
              default zero;
              description
                "Indicates whether the padding added to the
                 TWAMP-Test (UDP) packets will contain pseudo-random
                 numbers, or whether it should consist of all zeroes,
                 as per Section 4.2.1 of RFC 5357.";
            }

            leaf number-of-packets {
              type uint32;
              mandatory true;
              description
                "The overall number of TWAMP-Test (UDP) packets to be
                 transmitted by the Session-Sender for this test
                 session.";
            }
```

```
            choice packet-distribution {
              description
                "Indicates the distribution to be used for transmitting
                 the TWAMP-Test (UDP) packets.";
              case periodic {
                leaf periodic-interval {
                  type decimal64 {
                    fraction-digits 5;
                  }
                  units seconds;
                  mandatory true;
                  description
                    "Indicates the time to wait (in seconds) between
                     the first bits of TWAMP-Test (UDP) packet
                     transmissions for this test session.";
                  reference
                    "RFC 3432: Network performance measurement
                               with periodic streams";
                }
              }
              case poisson {
                leaf lambda {
                  type decimal64 {
                    fraction-digits 5;
                  }
                  units seconds;
                  mandatory true;
                  description
                    "Indicates the average time interval (in seconds)
                     between packets in the Poisson distribution.
                     The packet is calculated using the reciprocal of
                     lambda and the TWAMP-Test packet size (which
                     depends on the selected Mode and the packet
                     padding).";
                 reference
                   "RFC 2330: Framework for IP Performance Metrics";
                }
                leaf max-interval {
                  type decimal64 {
                    fraction-digits 5;
                  }
                  units seconds;
                  description
                    "Indicates the maximum time (in seconds)
                     between packet transmissions.";
                  reference
                    "RFC 7312: Advanced Stream and Sampling Framework
                               for IP Performance Metrics (IPPM)";
```

```
              }
            }
          }

          leaf state {
            type sender-session-state;
            config false;
            description
              "Indicates the Session-Sender test session state.";
          }

          uses maintenance-statistics;
        }
      }

      container session-reflector {
        if-feature session-reflector;
        description
          "Configuration of the TWAMP Session-Reflector logical
           entity";

        leaf admin-state {
          type boolean;
          default true;
          description
            "Indicates whether the device is allowed to operate
             as a TWAMP Session-Reflector.";
        }

        leaf refwait {
          type uint32 {
            range 1..604800;
          }
          units seconds;
          default 900;
          description
            "The Session-Reflector MAY discontinue any session that
             has been started when no packet associated with that
             session has been received for REFWAIT seconds. As per
             Section 3.1 of RFC 5357, this timeout allows a
             Session-Reflector to free up resources in case of
             failure.";
        }

        list test-session {
          key
            "sender-ip sender-udp-port
             reflector-ip reflector-udp-port";
```

```
            config false;
            description
              "TWAMP Session-Reflectortest sessions.";

            leaf sid {
              type string;
              description
                "An auto-allocated identifier for this TWAMP-Test
                 session that is unique within the context of this
                 Server/Session-Reflector device only. This value
                 is communicated to the Control-Client that
                 requested the test session in the SID field of the
                 Accept-Session message.";
            }

            leaf sender-ip {
              type inet:ip-address;
              description
                "The IP address on the remote device, which is the
                 source IP address used in the TWAMP-Test (UDP) packets
                 belonging to this test session.";
            }

            leaf sender-udp-port {
              type dynamic-port-number;
              description
                "The source UDP port used in the TWAMP-Test packets
                 belonging to this test session.";
            }

            leaf reflector-ip {
              type inet:ip-address;
              description
                "The IP address of the local Session-Reflector
                 device, which is the destination IP address used
                 in the TWAMP-Test (UDP) packets belonging to this test
                 session.";
            }

            leaf reflector-udp-port {
              type inet:port-number {
                range "862 | 49152..65535";
                }
              description
                "The destination UDP port number used in the
                 TWAMP-Test (UDP) test packets belonging to this
                 test session.";
            }
```

```
          leaf parent-connection-client-ip {
            type inet:ip-address;
            description
              "The IP address on the Control-Client device, which
               is the source IP address used in the TWAMP-Control
               (TCP) packets belonging to the parent control
               connection that negotiated this test session.";
          }

          leaf parent-connection-client-tcp-port {
            type inet:port-number;
            description
              "The source TCP port number used in the TWAMP-Control
               (TCP) packets belonging to the parent control
               connection that negotiated this test session.";
          }

          leaf parent-connection-server-ip {
            type inet:ip-address;
            description
              "The IP address of the Server device, which is the
               destination IP address used in the TWAMP-Control
               (TCP) packets belonging to the parent control
               connection that negotiated this test session.";
          }

          leaf parent-connection-server-tcp-port {
            type inet:port-number;
            description
              "The destination TCP port number used in the
               TWAMP-Control (TCP) packets belonging to the parent
               control connection that negotiated this test
               session.";
          }

          leaf test-packet-dscp {
            type inet:dscp;
            description
              "The DSCP value present in the IP header of
               TWAMP-Test (UDP) packets belonging to this session.";
          }

          uses maintenance-statistics;
        }
      }
    }
  }
```

<CODE ENDS>

6. Data Model Examples

This section presents a simple but complete example of configuring all four entities in Figure 1, based on the YANG module specified in Section 5. The example is illustrative in nature, but aims to be self-contained, i.e. were it to be executed in a real TWAMP implementation it would lead to a correctly configured test session. For completeness, examples are provided for both IPv4 and IPv6.

A more elaborated example, which also includes authentication parameters, is provided in Appendix A.

6.1. Control-Client

Figure 8 shows a configuration example for a Control-Client with client/admin-state enabled. In a real implementation following Figure 2 this would permit the initiation of TWAMP-Control connections and TWAMP-Test sessions.

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <client>
      <admin-state>true</admin-state>
    </client>
  </twamp>
</config>
```

Figure 8: XML instance enabling Control-Client operation.

The following example shows a Control-Client with two instances of client/ctrl-connection, one called "RouterA" and another called "RouterB". Each TWAMP-Control connection is to a different Server. The control connection named "RouterA" has two test session requests. The TWAMP-Control connection named "RouterB" has no TWAMP-Test session requests.

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <client>
```

```
        <admin-state>true</admin-state>
        <ctrl-connection>
          <name>RouterA</name>
          <client-ip>203.0.113.1</client-ip>
          <server-ip>203.0.113.2</server-ip>
          <test-session-request>
            <name>Test1</name>
            <sender-ip>203.0.113.3</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>203.0.113.4</reflector-ip>
            <reflector-udp-port>50001</reflector-udp-port>
            <start-time>0</start-time>
          </test-session-request>
          <test-session-request>
            <name>Test2</name>
            <sender-ip>203.0.113.1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>203.0.113.2</reflector-ip>
            <reflector-udp-port>50001</reflector-udp-port>
            <start-time>0</start-time>
          </test-session-request>
        </ctrl-connection>
        <ctrl-connection>
          <name>RouterB</name>
          <client-ip>203.0.113.1</client-ip>
          <server-ip>203.0.113.3</server-ip>
        </ctrl-connection>
      </client>
    </twamp>
  </config>

  [note: '\' line wrapping is for formatting only]

  <?xml version="1.0" encoding="utf-8"?>
  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <client>
        <admin-state>true</admin-state>
        <ctrl-connection>
          <name>RouterA</name>
          <client-ip>2001:DB8:203:0:113::1</client-ip>
          <server-ip>2001:DB8:203:0:113::2</server-ip>
          <test-session-request>
            <name>Test1</name>
            <sender-ip>2001:DB8:203:1:113::3</sender-ip>
            <sender-udp-port>54000</sender-udp-port>
            <reflector-ip>2001:DB8:203:1:113::4</reflector-ip>
            <reflector-udp-port>55000</reflector-udp-port>
```

```
            <start-time>0</start-time>
          </test-session-request>
          <test-session-request>
            <name>Test2</name>
            <sender-ip>2001:DB8:203:0:113::1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
            <reflector-udp-port>55001</reflector-udp-port>
            <start-time>0</start-time>
          </test-session-request>
        </ctrl-connection>
        <ctrl-connection>
          <name>RouterB</name>
          <client-ip>2001:DB8:203:0:113::1</client-ip>
          <server-ip>2001:DB8:203:0:113::3</server-ip>
        </ctrl-connection>
      </client>
    </twamp>
  </config>
```

6.2. Server

Figure 9 shows a configuration example for a Server with server/admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Control connections and TWAMP-Test sessions.

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <server>
      <admin-state>true</admin-state>
    </server>
  </twamp>
</config>
```

Figure 9: XML instance enabling Server operation.

The following example presents a Server with the TWAMP-Control connection corresponding to the control connection name (client/ctrl-connection/name) "RouterA" presented in Section 6.1.

[note: '\' line wrapping is for formatting only]

```
<?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>
```

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <server>
      <admin-state>true</admin-state>
      <ctrl-connection>
        <client-ip>2001:DB8:203:0:113::1</client-ip>
        <client-tcp-port>16341</client-tcp-port>
        <server-ip>2001:DB8:203:0:113::2</server-ip>
        <server-tcp-port>862</server-tcp-port>
        <state>active</state>
      </ctrl-connection>
    </server>
  </twamp>
</data>
```

6.3. Session-Sender

Figure 10 shows a configuration example for a Session-Sender with session-sender/admin-state enabled, which permits a device following Figure 2 to initiate TWAMP-Test sessions.

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-sender>
      <admin-state>true</admin-state>
    </session-sender>
  </twamp>
</config>
```

Figure 10: XML instance enabling Session-Sender operation.

The following configuration example shows a Session-Sender with the two TWAMP-Test sessions presented in Section 6.1.

[note: '\' line wrapping is for formatting only]

```
<?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>
      </test-session>
      <test-session>
        <name>Test2</name>
        <ctrl-connection-name>RouterA</ctrl-connection-name>
        <number-of-packets>900</number-of-packets>
        <lambda>1</lambda>
        <max-interval>2</max-interval>
      </test-session>
    </session-sender>
  </twamp>
</data>
```

6.4. Session-Reflector

This configuration example shows a Session-Reflector with session-reflector/admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Test sessions.

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
    </session-reflector>
  </twamp>
</config>
```

Figure 11: XML instance enabling Session-Reflector operation.

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

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
      <test-session>
        <sender-ip>203.0.113.3</sender-ip>
        <sender-udp-port>54000</sender-udp-port>
        <reflector-ip>203.0.113.4</reflector-ip>
        <reflector-udp-port>50001</reflector-udp-port>
        <sid>1232</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <sent-packets>2</sent-packets>
        <rcv-packets>2</rcv-packets>
        <last-sent-seq>1</last-sent-seq>
        <last-rcv-seq>1</last-rcv-seq>
      </test-session>
      <test-session>
        <sender-ip>203.0.113.1</sender-ip>
        <sender-udp-port>54001</sender-udp-port>
        <reflector-ip>192.68.0.2</reflector-ip>
        <reflector-udp-port>50001</reflector-udp-port>
```

```
          <sid>178943</sid>
          <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
          <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
rver-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>
```

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
      <test-session>
        <sender-ip>203.0.113.3</sender-ip>
        <sender-udp-port>54000</sender-udp-port>
        <reflector-ip>203.0.113.4</reflector-ip>
        <reflector-udp-port>54001</reflector-udp-port>
        <sid>1232</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <sent-packets>2</sent-packets>
        <rcv-packets>2</rcv-packets>
        <last-sent-seq>1</last-sent-seq>
        <last-rcv-seq>1</last-rcv-seq>
      </test-session>
      <test-session>
        <sender-ip>203.0.113.1</sender-ip>
        <sender-udp-port>54001</sender-udp-port>
        <reflector-ip>192.68.0.2</reflector-ip>
```

```
          <reflector-udp-port>55001</reflector-udp-port>
          <sid>178943</sid>
          <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
          <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
rver-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 specified in Section 5 this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF [RFC6241] layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF Access Control Module (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content..

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 IETF XML Registry [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 YANG [RFC6020].

name: ietf-twamp

namespace: urn:ietf:params:xml:ns:yang:ietf-twamp

prefix: twamp

reference: RFC XXXX

9. Acknowledgements

We thank Fred Baker, Kevin D'Souza, Gregory Mirsky, Brian Trammell, Robert Sherman, and Marius Georgescu for their thorough and constructive reviews, comments and text suggestions.

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

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

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

10. Contributors

Lianshu Zheng.

11. References

11.1. Normative References

[I-D.ietf-ippm-port-twamp-test]
Morton, A. and G. Mirsky, "OWAMP and TWAMP Well-Known Port Assignments", draft-ietf-ippm-port-twamp-test-01 (work in progress), March 2018.

[RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, DOI 10.17487/RFC1305, March 1992, <https://www.rfc-editor.org/info/rfc1305>.

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

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://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, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6038>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7312] Fabini, J. and A. Morton, "Advanced Stream and Sampling Framework for IP Performance Metrics (IPPM)", RFC 7312, DOI 10.17487/RFC7312, August 2014, <https://www.rfc-editor.org/info/rfc7312>.

[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, <https://www.rfc-editor.org/info/rfc7717>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

[I-D.ietf-ippm-metric-registry]
Bagnulo, M., Claise, B., Eardley, P., Morton, A., and A. Akhter, "Registry for Performance Metrics", draft-ietf-ippm-metric-registry-14 (work in progress), March 2018.

[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.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[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, <https://www.rfc-editor.org/info/rfc7426>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

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

[note: '\' line wrapping is for formatting only]

```
<?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>c2VjcmV0MQ==</secret-key>
      </key-chain>
      <key-chain>
        <key-id>KeyForRouterB</key-id>
        <secret-key>c2VjcmV0Mg0K</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>
        <control-packet-dscp>32</control-packet-dscp>
        <key-id>KeyClient1ToRouterA</key-id>
        <test-session-request>
          <name>Test1</name>
          <sender-ip>203.0.113.3</sender-ip>
          <sender-udp-port>54000</sender-udp-port>
          <reflector-ip>203.0.113.4</reflector-ip>
          <reflector-udp-port>55000</reflector-udp-port>
```

```
            <padding-length>64</padding-length>
            <start-time>0</start-time>
          </test-session-request>
          <test-session-request>
            <name>Test2</name>
            <sender-ip>203.0.113.1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>203.0.113.2</reflector-ip>
            <reflector-udp-port>55001</reflector-udp-port>
            <padding-length>128</padding-length>
            <start-time>0</start-time>
          </test-session-request>
        </ctrl-connection>
      </client>
    </twamp>
  </data>
```

[note: '\' line wrapping is for formatting only]

```
<?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>c2VjcmV0MQ==</secret-key>
      </key-chain>
      <key-chain>
        <key-id>KeyForRouterB</key-id>
        <secret-key>c2VjcmV0Mg0K</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>
        <control-packet-dscp>32</control-packet-dscp>
        <key-id>KeyClient1ToRouterA</key-id>
        <test-session-request>
          <name>Test1</name>
```

```
            <sender-ip>2001:DB8:10:1:1::1</sender-ip>
            <sender-udp-port>54000</sender-udp-port>
            <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
            <reflector-udp-port>55000</reflector-udp-port>
            <padding-length>64</padding-length>
            <start-time>0</start-time>
          </test-session-request>
          <test-session-request>
            <name>Test2</name>
            <sender-ip>2001:DB8:203:0:113::1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
            <reflector-udp-port>55001</reflector-udp-port>
            <padding-length>128</padding-length>
            <start-time>0</start-time>
          </test-session-request>
        </ctrl-connection>
      </client>
    </twamp>
  </data>
```

A.2. Server

```
  [note: '\' line wrapping is for formatting only]

  <?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>
        <control-packet-dscp>32</control-packet-dscp>
        <modes>authenticated unauthenticated</modes>
        <count>30</count>
        <key-chain>
          <key-id>KeyClient1ToRouterA</key-id>
          <secret-key>c2VjcmV0MQ==</secret-key>
        </key-chain>
        <key-chain>
          <key-id>KeyClient10ToRouterA</key-id>
          <secret-key>c2VjcmV0MTANCg==</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>
```

```
          <control-packet-dscp>32</control-packet-dscp>
          <selected-mode>unauthenticated</selected-mode>
          <key-id>KeyClient1ToRouterA</key-id>
          <count>30</count>
        </ctrl-connection>
      </server>
    </twamp>
  </data>
```

[note: '\' line wrapping is for formatting only]

```
<?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>
      <control-packet-dscp>32</control-packet-dscp>
      <modes>authenticated unauthenticated</modes>
      <count>30</count>
      <key-chain>
        <key-id>KeyClient1ToRouterA</key-id>
        <secret-key>c2VjcmV0MQ==</secret-key>
      </key-chain>
      <key-chain>
        <key-id>KeyClient10ToRouterA</key-id>
        <secret-key>c2VjcmV0MTANCg==</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>
        <control-packet-dscp>32</control-packet-dscp>
        <selected-mode>unauthenticated</selected-mode>
        <key-id>KeyClient1ToRouterA</key-id>
        <count>30</count>
      </ctrl-connection>
    </server>
  </twamp>
</data>
```

A.3. Session-Sender

[note: '\' line wrapping is for formatting only]

```
<?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>
        <sent-packets>2</sent-packets>
        <rcv-packets>2</rcv-packets>
        <last-sent-seq>1</last-sent-seq>
        <last-rcv-seq>1</last-rcv-seq>
      </test-session>
      <test-session>
        <name>Test2</name>
        <ctrl-connection-name>RouterA</ctrl-connection-name>
        <fill-mode>random</fill-mode>
        <number-of-packets>900</number-of-packets>
        <lambda>1</lambda>
        <max-interval>2</max-interval>
        <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

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
      <test-session>
        <sender-ip>203.0.113.3</sender-ip>
        <sender-udp-port>54000</sender-udp-port>
        <reflector-ip>203.0.113.4</reflector-ip>
```

```
          <reflector-udp-port>55000</reflector-udp-port>
          <sid>1232</sid>
          <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
          <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
          <test-packet-dscp>32</test-packet-dscp>
          <sent-packets>2</sent-packets>
          <rcv-packets>2</rcv-packets>
          <last-sent-seq>1</last-sent-seq>
          <last-rcv-seq>1</last-rcv-seq>
        </test-session>
        <test-session>
          <sender-ip>203.0.113.1</sender-ip>
          <sender-udp-port>54001</sender-udp-port>
          <reflector-ip>192.68.0.2</reflector-ip>
          <reflector-udp-port>55001</reflector-udp-port>
          <sid>178943</sid>
          <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
          <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
          <test-packet-dscp>32</test-packet-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>

  [note: '\' line wrapping is for formatting only]

  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <session-reflector>
        <admin-state>true</admin-state>
        <test-session>
```

```
          <sender-ip>2001:DB8:10:1:1::1</sender-ip>
          <sender-udp-port>54000</sender-udp-port>
          <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
          <reflector-udp-port>55000</reflector-udp-port>
          <sid>1232</sid>
          <parent-connection-client-ip>2001:DB8:203:0:113::1</parent-c\
onnection-client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
          <parent-connection-server-ip>2001:DB8:203:0:113::2</parent-c\
onnection-server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
          <test-packet-dscp>32</test-packet-dscp>
          <sent-packets>2</sent-packets>
          <rcv-packets>2</rcv-packets>
          <last-sent-seq>1</last-sent-seq>
          <last-rcv-seq>1</last-rcv-seq>
        </test-session>
        <test-session>
          <sender-ip>2001:DB8:203:0:113::1</sender-ip>
          <sender-udp-port>54001</sender-udp-port>
          <reflector-ip>2001:DB8:192:68::2</reflector-ip>
          <reflector-udp-port>55001</reflector-udp-port>
          <sid>178943</sid>
          <parent-connection-client-ip>2001:DB8:203:0:113::1</parent-c\
onnection-client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
          <parent-connection-server-ip>2001:DB8:203:0:113::2</parent-c\
onnection-server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
          <test-packet-dscp>32</test-packet-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, TWAMP [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

Email: mjethanandani@gmail.com

Kostas Pentikousis (editor)
Travelping
Siemensdamm 50
Berlin 13629
Germany

Email: k.pentikousis@travelping.com