This document specifies the data model for implementations of Session-Sender and Session-Reflector for Two-Way Active Measurement Protocol (TWAMP) Light mode using YANG.
1. Introduction

The Two-Way Active Measurement Protocol (TWAMP) [RFC5357] can be used to measure performance parameters of IP networks such as latency, jitter, and packet loss by sending test packets and monitoring their experience in the network. The [RFC5357] defines two protocols, TWAMP Control and TWAMP Test, and a profile of TWAMP Test, TWAMP Light. The TWAMP Light is known to have many implementations though no common management framework being defined, thus leaving some aspects of test packet processing to interpretation. The goal of this document is to collect analyze these variations; describe common model while allowing for extensions in the future. This document defines such a TWAMP data model and specifies it formally using the YANG data modeling language [RFC6020].

1.1. Conventions used in this document

1.1.1. Requirements Language

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

The scope of this document includes model of the TWAMP Light as defined in Appendix I of [RFC5357] as well as model of accepted Errata. The former mode of TWAMP Light will be referred in this document as Stateless and the latter - Stateful. This document benefits from earlier attempt to define TWAMP MIB in [I-D.elteto-ippm-twamp-mib] and from TWAMP YANG model defined in [I-D.cmzrjp-ippm-twamp-yang].

Figure 1 updates TWAMP-Light reference model presented in Appendix I [RFC5357] for the scenario when instantiation of a TWAMP-Test session between Session-Sender and Session-Reflector controlled by communication between a Configuration Client as a manager and Configuration Servers as agents of the configuration session.

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

Figure 1: TWAMP Light Reference Model

2.1. Data Model Parameters

2.2. Session-Sender

TBA

2.3. Session-Reflector

TBA

3. Data Model

Creating TWAMP-Light data model presents number of challenges and among them is identification of a test-session at Session-Reflector. A Session-Reflector MAY require only as little as its IP and UDP port
number in received TWAMP-Test packet to spawn new test session. More so, to test processing of Class-of-Service along the same route in Equal Cost Multi-Path environment Session-Sender may run TWAMP test sessions concurrently using the same source IP address, source UDP port number, destination IP address, and destination UDP port number. Thus the only parameter that can be used to differentiate these test sessions would be DSCP value. The DSCP field may get re-marked along the path and without use of [RFC7750] that will go undetected, but by using five-tuple instead of four-tuple as a key we can ensure that TWAMP test packets that are considered as different test sessions follow the same path even in ECMP environments.

3.1. Tree Diagram
module: ietf-twamp-light

module twampLightSessionSender {sessionSenderLight)?
  +--rw testSession* [senderIp senderUdpPort reflectorIp
                           reflectorUdpPort dscp]
      +--rw numOfPackets? uint32
      +--rw packetPaddingSize? uint32
      +--rw sessionAuthenticationMode? enumeration
      +--rw interval? uint32
      +--ro senderSessionState? enumeration
      +--ro sentPackets? uint32
      +--ro rcvPackets? uint32
      +--ro lastSentSeq? uint32
      +--ro lastRcvSeq? uint32
      +--rw senderIp inet:ip-address
      +--rw senderUdpPort inet:port-number
      +--rw reflectorIp inet:ip-address
      +--rw reflectorUdpPort inet:port-number
      +--rw dscp inet:dscp

module twampLightSessionReflector {sessionReflectorLight)?
  +--rw reflectorLightState boolean
  +--rw refwait? uint32
  +--rw reflectorLightMode? enumeration
  +--rw dscpHandlingMode? enumeration
  +--rw testSession* [senderIp senderUdpPort reflectorIp
                       reflectorUdpPort dscp]
      +--ro sentPackets? uint32
      +--ro rcvPackets? uint32
      +--ro lastSentSeq? uint32
      +--ro lastRcvSeq? uint32
      +--rw senderIp inet:ip-address
      +--rw senderUdpPort inet:port-number
      +--rw reflectorIp inet:ip-address
      +--rw reflectorUdpPort inet:port-number
      +--rw dscp inet:dscp

3.2. YANG Module

<CODE BEGINS> file "ietf-twamp-light@2016-0305"
module ietf-twamp-light {
    //namespace need to be assigned by IANA
    prefix ietf-twamp-light;

    import ietf-inet-types {
        prefix inet;
    }
}
organization
  "IETF IPPM (IP Performance Metrics) Working Group";

contact
  "draft-mirsky-ippm-twamp-light-yang@tools.ietf.org";

description "TWAMP Light Data Model";

revision "2016-03-05" {
  description "01 version. RFC5357 is covered, including Appendix I and the Errata.";
  reference "draft-mirsky-ippm-twamp-light-yang";
}

feature sessionSenderLight {
  description "This feature relates to the device functions as the TWAMP Light Session-Sender.";
}

feature sessionReflectorLight {
  description "This feature relates to the device functions as the TWAMP Light Session-Reflector.";
}

grouping maintenanceStatistics {
  description "Maintenance statistics grouping";
  leaf sentPackets {
    type uint32;
    default 0;
    config "false";
    description "Packets sent";
  }
  leaf rcvPackets {
    type uint32;
    default 0;
    config "false";
    description "Packets received";
  }
  leaf lastSentSeq {
    type uint32;
    default 0;
    config "false";
    description "Last sent sequence number";
  }
  leaf lastRcvSeq {
    type uint32;
    default 0;
    config "false";
description "Last received sequence number";
}
}

grouping sessionLightParameters {
  description "Parameters common among Session-Sender and
  Session-Reflector.";
  leaf senderIp {
    type inet:ip-address;
    description "Sender IP address";
  }
  leaf senderUdpPort {
    type inet:port-number {
      range "49152..65535";
      description "Sender UDP port number";
    }
  }
  leaf reflectorIp {
    type inet:ip-address;
    description "Reflector IP address";
  }
  leaf reflectorUdpPort {
    type inet:port-number {
      range "49152..65535";
      description "Reflector UDP port number";
    }
  }
  leaf dscp {
    type inet:dscp;
    description "The DSCP value to be placed in the header of TWAMP
    UDP
    test packets generated by the Session-Sender. Whether
    Session-Reflector uses this value depends upon its local
    configuration.";
  }
}

container twampLightSessionSender {
  if-feature sessionSenderLight;
  description "TWAMP-Light Session-Sender container";
  list testSession {
    key "senderIp senderUdpPort reflectorIp reflectorUdpPort dscp";
    ordered-by system;
    description "This structure is a container of test session
    managed objects.";
    leaf numberOfPackets {
      type uint32;
      description "The overall number of UDP test packets to be
      transmitted by the sender for this test session.";
    }
  }
}
leaf packetPaddingSize {
  type uint32;
  default 27;
  description "Size of the Packet Padding. Suggested to run Path MTU Discovery to avoid packet fragmentation in IPv4 and packet backholing in IPv6.";
}

leaf sessionAuthenticationMode {
  type enumeration {
    enum unauthenticated {
      description "Unauthenticated TWAMP-Light test session";
    }
    enum authenticated {
      description "Authenticated TWAMP-Light test session";
    }
    enum encrypted {
      description "Encrypted TWAMP-Light test session";
    }
  }
  default unauthenticated;
  description "Authentication mode of the TWAMP-Light test session."
}

leaf interval {
  type uint32;
  description "Time interval between transmission of two consecutive packets in the test session.";
}

leaf senderSessionState {
  type enumeration {
    enum active {
      description "Test session is active.";
    }
    enum ready {
      description "Test session is idle.";
    }
  }
  default ready;
  config "false";
  description "State of the particular TWAMP-Light test
session at the sender.
);
}
uses maintenanceStatistics;
uses sessionLightParameters;
}
}

container twampLightSessionReflector {
  if-feature sessionReflectorLight;
  description "TWAMP-Light Session-Reflector container";
  leaf reflectorLightState {
    type boolean;
    mandatory "true";
    description "Whether this network element is enabled to
    act as TWAMP-Light Reflector";
  }

  leaf refwait {
    type uint32 {
      range 1..604800;
    }
    units seconds;
    default 900;
    description "REFWAIT(TWAMP test session timeout in seconds),
    the default value is 900";
  }

  leaf reflectorLightMode {
    type enumeration {
      enum stateful {
        description "When the Session-Reflector Light is stateful,
        i.e. is aware of test session state.";
      }
      enum stateless {
        description "when the Session-Reflector is stateless.";
      }
    }
    default stateless;
    description "Whether Session-Sender copies sequence number
    of received TWAMP-Test packet, i.e. Stateless, or counts
    reflected TWAMP-Test packets and restarts counter based on
    external event.";
  }

  leaf dscpHandlingMode {
    type enumeration {
      enum copyReceivedValue {
        description "Use DSCP value copied from received TWAMP
test packet of the test session.

enum useConfiguredValue {
    description "Use DSCP value configured for this test
    session on the Session-Reflector."
}

default copyReceivedValue;

description "Session-Reflector handling of DSCP:
    - use value copied from received TWAMP-Test packet;
    - use value explicitly configured."

list testSession {
    key "senderIp senderUdpPort reflectorIp reflectorUdpPort dscp";
    description "This structure is a container of test session
    managed objects."
    uses maintenanceStatistics;
    uses sessionLightParameters;
}

4. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.


Registrant Contact: The IPPM WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-twamp-light


prefix: twamp

reference: RFC XXXX
5. Security Considerations

The configuration, state, action data defined in this document may be accessed via the NETCONF protocol [RFC6241]. SSH [RFC6242] is mandatory secure transport that is the lowest NETCONF layer. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

But, in general, this TWAMP Light YANG module does not change any underlying security issues that already may exist in [I-D.elteto-ippm-twamp-mib].

6. Acknowledgements

7. References

7.1. Normative References

[I-D.cmzrjp-ippm-twamp-yang]

[I-D.elteto-ippm-twamp-mib]


7.2. Informative References


Authors’ Addresses

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Tamas Elteto
Ericsson

Email: tamas.elteto@ericsson.com