NETMOD Working Group   N. Sambo
Internet-Draft   P. Castoldi
Intended status: Standards Track   Scuola Superiore Sant'Anna
Expires: January 3, 2019   G. Fioccola
                                                         Telecom Italia
                                                             F. Cugini
                                                                  CNIT
                                                               H. Song
                                                               T. Zhou
                                                                Huawei
                                                          July 2, 2018

YANG model for finite state machine
draft-sambo-netmod-yang-fsm-03

Abstract

   Network operators and service providers are facing the challenge of
   deploying systems from different vendors while looking for a trade-
   off among transmission performance, network device reuse, and capital
   expenditure without the need of being tied to single vendor
   equipment. The deployment and operation of more dynamic and
   programmable network infrastructures can be driven by adopting model-
   driven and software-defined control and management paradigms. In
   this context, YANG enables to compile a set of consistent vendor-
   neutral data models for networks and components based on actual
   operational needs emerging from heterogeneous use cases. This
   document proposes YANG models to describe events, operations, and
   finite state machine of YANG-defined network elements. The proposed
   models can be applied in several use cases: i) in the context of
   optical networks to pre-instruct data plane devices (e.g., an optical
   transponder) on the actions to be performed (e.g., code adaptation)
   in case some events, such as physical layer degradations, occur; ii)
   in general data networks, network telemetry applications can define
   and embed custom data probes into data plane devices. A probe in
   many cases can be modeled as an FSM; iii) the monitoring of packet
   loss and delay through a network clustering approach.

Status of This Memo

Table of Contents

1. Introduction

Networks are evolving toward more programmability, flexibility, and
multi-vendor interoperability. Multi-vendor interoperability can be
applied in the context of nodes, i.e. a node composed of components
provided by different vendors (named fully disaggregated white box)
is assembled under the same control system. This way, operators can
optimize costs and network performance without the need of being tied
to single vendor equipment. NETCONF protocol RFC6241 [RFC6241] based
on YANG data modeling language RFC6020 [RFC6020] is emerging as a
candidate Software Defined Networking (SDN) enabled protocol. First,
NETCONF supports both control and management functionalities, thus
permits high programmability. Then, YANG enables data modeling in a
vendor-neutral way. Some recent works have provided YANG models to
describe attributes of links (e.g., identification), nodes (e.g.,
connectivity matrix), media channels, and transponders (e.g.,
supported forward error correction - FEC) of networks
([I-D.ietf-i2rs-yang-network-topo] [I-D.vergara-ccamp-flexigrid-yang]
[I-D.zhang-ccamp-l1-topo-yang]), also including optical technologies.
This document presents YANG models to describe events, operations,
and finite state machine of YANG-defined network elements. Such
models can be applied to several use cases. In the context of
elastic optical networks (EONs), the model enables a centralized
remote network controller (managed by a network operator) to instruct
a transponder controller about the actions to perform when certain
events (e.g., failures) occur. The actions to be taken and the
events can be re-programmed on the device. In general data networks,
programmable network telemetry is considered a killer SDN application
which can help applications gain unprecedented visibility to network
data plane. Instead of providing raw data, network devices can be
configured to filter and process data directly on the data plane and
only hand preprocessed data to the collector, in order to save data
bandwidth and reduce reaction delay ([I-D.song-opsawg-dnp4iq]) . Such
configurations can be programed as custom probes and dynamically
deployed into data plane devices. A probe in many cases can be
modeled as an FSM. Another use case is the monitoring of packet loss
and delay through a network clustering approach: in this case, each
FSM state is determined by a specific subdivision of the network in
Clusters ([I-D.fioccola-ippm-multipoint-alt-mark]).

2. Conventions used in this document

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

[3](). Terminology

ABNO: Application-Based Network Operations

BER: Bit Error Rate

EON: Elastic Optical Network

FEC: Forward Error Correction

FSM: Finite State Machine

NETCONF: Network Configuration Protocol

OAM: Operation Administration and Maintenance

SDN: Software Defined Network

YANG: Yet Another Network Generator

DNP: Dynamic Network Probe

AMM: Alternate Marking Method

[4](). Example of application

[4.1](). Pre-programming resiliency schemes in EONs

EONs (optical networks based on flexible grid supporting circuits of
different bandwidth) are expected to employ flexible transponders,
i.e. transponders supporting multiple bit rates, multiple modulation
formats, and multiple codes. Such transponders permits the (re-)
configuration of the bit rate value based on traffic requirements, as
well as the configuration of the modulation format and code based on
the physical characteristics of a path (e.g., quadrature phase shift
keying is more robust than 16 quadrature amplitude modulation). This
way, transmission parameters can be (re-) configured based on
physical layer changes. The YANG model presented in this draft
enables to pre-program reconfiguration settings of data plane devices
in case of failures or physical layer degradations. In particular,
soft failures are assumed. Soft failures imply transmission
performance degradation, in turns a bit error rate (BER) increase,
e.g. due to the ageing of some network devices. Without loosing
generality, the ABNO architecture is assumed for the control and
management of EONs ([RFC7491]() [[RFC7491]()]). Considering the state of the
art, when pre-FEC BER passes above a predefined threshold, it is
expected that an alarm is sent to the OAM Handler, which communicates
with the ABNO controller that may trigger an SDN controller (that

could be the Provisioning Manager of ABNO RFC7491 [RFC7491]) for
computing new transmission parameters. The involved ABNO modules are
shown in the simplified ABNO architecture of Fig. 1. Then,
transponders are reconfigured. When alarms related to several
connections impacted by the soft failure are generated, this
procedure may be particularly time consuming. The related workflow
for transponder reconfiguration is shown in Fig. 2. The proposed
model enables an SDN controller to instruct the transponder about
reconfiguration of new transmission parameters values if a soft
failure occurs. This can be done before the failure occurs (e.g.,
during the connection instantiation phase or during the connection
service), so that data plane devices can promptly reconfigure
themselves without querying the SDN controller to trigger an on-
demand recovery. This is expected to speed up the recovery process
from soft failures. The related flow chart is shown in Fig. 3. The
whole mechanism is based on a finite state machine where each state
is associated to a specific configuration of transmission parameters
(e.g., modulation format). The transition from a state to another
state is triggered by specific events at the physical layer such as
the bit error rate above a threshold. The transition from a state to
another state implies a set of actions, including the change of
transmission parameters (e.g., modulation format), which are actually
suitable for the current condition at the physical layer. Moreover,
since transmission and receiver must be synchronized about the
transmission settings (modulation format and so no) for a proper
transmission, another action consists of this synchronization. Thus,
when the transponder at the receiver side decides to change its
transmission parameters based on the monitored BER, the remote
transponder at the transmitter side has to do the same state
transition. In particular, the transponder at the receiver side
sends a message to the transmitter to synchronize about the
transmission parameters to be adopted. This message can be sent over
a control channel. This way both the transmitter and receiver
operates with the same transmission parameters: e.g. the format, FEC,
and so on. No central controller is involved at this stage, only a
notification can be sent to the central controller to inform it about
the successful reconfiguration.

```
         _____              _____
        |  ABNO       |            |   OAM       |
        |controller   |   ------   |  Handler    |
        |_____|            |_____|

             |                           |
             |                           |
             |                           |
         _____                   |
        |   SDN       |                   |
        | controller  |                   |
        |_____|                   |
                                          |
             |                            |
             |                            |
             |                            |
          _____     |
        |          Client            |
        |          network           |
        |_____|
```
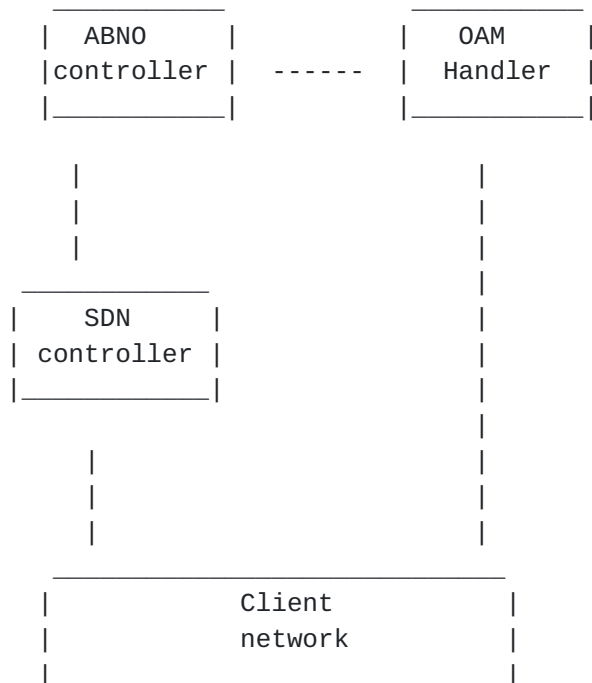
                Figure 1: Assumed ABNO functional modules

```
             _____
            |         1           |
            |Sending alarm to the |
            |     OAM Handler     |
            |                     |
            |_____|
                      |
                      |
                      |
             _____
            |         2           |
            |       Trigger       |
            |    SDN Controller   |
            |                     |
            |_____|
                      |
                      |
                      |
             _____
            |         3           |
            |    Computation of   |
            |   new transmission  |
            |      parameters     |
            |_____|
                      |
                      |
                      |
             _____
            |         4           |
            |     Data plane      |
            |    reconfiguration  |
            |                     |
            |_____|
```
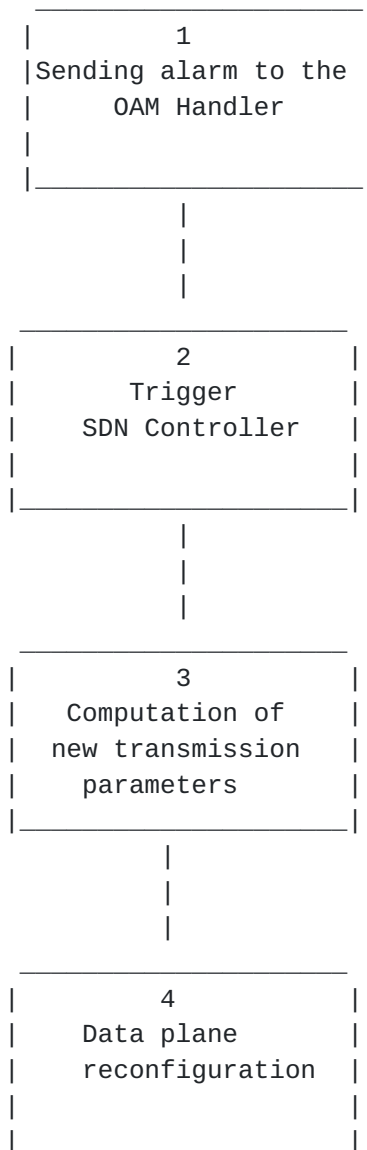
Figure 2: Flow chart of the expected state-of-the-art approach

```
      _____
     |            1           |
     | Instructing the local  |
     |      controller of     |
     |   data plane devices   |
     |_____|
                  |
                  |
                  |
      _____
     |            2           |
     | Local reconfiguration  |
     |      upon failure      |
     |        detection       |
     |_____|
                  |
                  |
                  |
      _____
     |            3           |
     |                        |
     |      notification      |
     |                        |
     |_____|
```

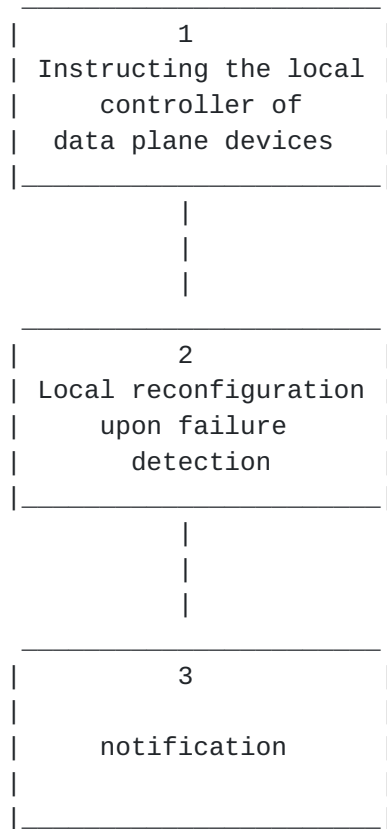   Figure 3: Flow chart of the approach exploiting YANG models in this
                                   draft

4.2. Deploying Dynamic Probes for Programmable Network Telemetry

   In the past, network data analytics was considered a separate
   function from networks. They consume raw data extracted from
   networks through piecemeal protocols and interfaces. With the advent
   of user programmable data plane, we expect a paradigm shift that
   makes the data plane be an active component of the data telemetry and
   analytics solution. The programmable in-network data preprocessing
   is efficient and flexible to offload some light-weight data
   processing through dynamic data plane programming or configuration.
   A universal network data analytics platform built on top of this
   enables a tight and agile network control and OAM feedback loop. A
   proposed dynamic network telemetry system architecture is illustrated
   in Fig.4.

   An application translates its data requirements into a set of Dynamic
   Network Probes (DNP) targeting a subset of data plane devices. After
   the probes are deployed, each probe conducts its corresponding in-
   network data preprocessing and feeds the preprocessed data to the

collector. The collector finishes the data post-processing and
presents the results to the data-requesting application.

```
                +------------------------------------+
                |  network telemetry applications    |
                +------------------------------------+
                     ^                    |
                     |                    V
                     |         +-------------------+
                     |         | DNP compile/config |
                     |         +-------------------+
                     |                 |
                     |                 V
           +---------------+ +-------------------+
           |data collection| | Probe deployment  |
           +---------------+ +-------------------+
             ^   ^   ^              |   |   |
             |   |   |              V   V   V
           +------------------------------------+
           |  network data plane devices        |
           | (in-network data preprocessing)    |
           +------------------------------------+
```
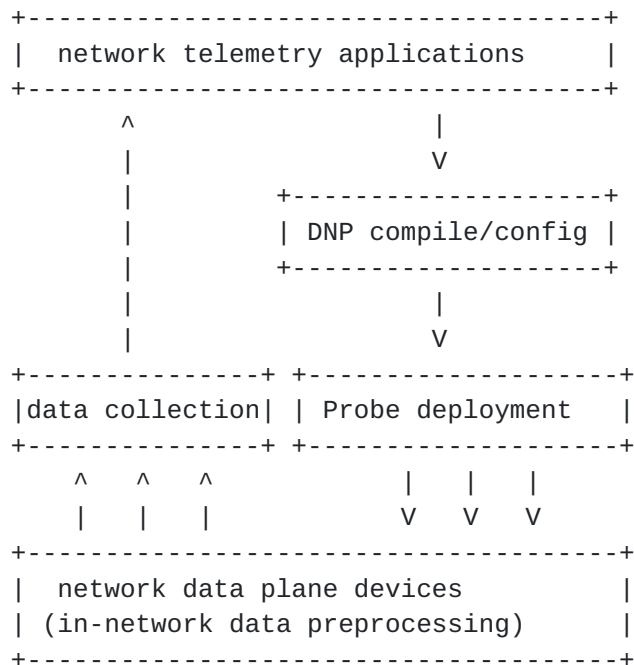
        Figure 4: Deploy dynamic network probes using YANG FSM models

   Many DNPs can be modeled as FSM which are configured to capture
   specific events. Here FSMs essentially preprocess the raw stream
   data and only report the necessary data to subscribing applications.

   For example, a congestion control application needs to monitor the
   router buffer occupancy. Instead of polling the buffer depth
   periodically, it is only interested in the real-time events when the
   buffer depth crosses a low and a high threshold. We can install a
   probe to achieve this data plane function and the probe can be
   modeled as a three-state FSM. Each state represents a buffer region:
   below the low threshold, above the high threshold, and in between the
   two thresholds. A possible state transition is checked against the
   buffer depth for each incoming and outgoing packet. Whenever a state
   transition happens, an event is generated and reported to the
   application. This approach significantly reduces the amount of data
   sent to the application and also allows a timely event notification.

   For another example, an application would like to monitor the delay
   experienced by a flow. The packet delay on its forwarding path can
   be acquired by using iOAM [I-D.brockners-inband-oam-requirements].
   However, the application only needs to know that N consecutive flow
   packets experience a delay longer than T. Instead of forwarding the

raw delay data to the application, a probe can be deployed to detect
the event. Similarly, the probe can be modeled as an FSM.

4.3. IP Performance Measurements on multipoint-to-multipoint large Networks

Networks offer rich sets of network performance measurement data, but
traditional approaches run into limitations. One reason for this is
the fact that in many cases, the bottleneck is the generation and
export of the data and the amount of data that can be reasonably
collected from the network runs into bandwidth and processing
constraints in the network itself. In addition, management tasks
related to determining and configuring which data to generate lead to
significant deployment challenges.

In order to address these issues, an SDN controller application
orchestrates network performance measurements tasks across the
network to allow an optimized monitoring. In fact the IP Performance
Measurement SDN Controller Application in Figure 5 can calibrate how
deep can be obtained monitoring data from the network by configuring
measurement points roughly or meticulously. This can be established
by using the feedback mechanism reported in Figure 5.

For instance, the SDN Controller can configure initially an end to
end monitoring between ingress points and egress points of the
network. If the network does not experiment issues, this approximate
monitoring is good enough and is very cheap in terms of network
resources. But, in case of problems, the SDN Controller becomes
aware of the issues from this approximate monitoring and, in order to
localize the portion of the network that has issues, configures the
measurement points more exhaustively. So a new detailed monitoring
is performed. After the detection and resolution of the problem the
initial approximate monitoring can be used again. This idea is
general and can be applied to different performance measurements
techniques both active and passive (and hybrid).

```
                +-------------------------------------+
                |       IP Performance Measurement    |
                |       SDN Controller Application    |
                +-------------------------------------+
                   ^   ^   ^             |   |   |
                   |   |   |             v   v   v
                +-------------------------------------+
                |           Multipoint Network        |
                +-------------------------------------+
```
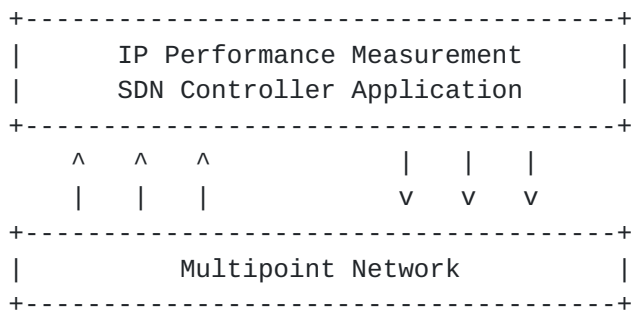
Figure 5: Feedback mechanism on multipoint-to-multipoint large
Networks

One of the most efficient methodology to perform packet, loss delay
and jitter measurements both in an IP and Overlay Networks is the
Alternate Marking method, as presented in [I-D.ietf-ippm-alt-mark]
and [I-D.fioccola-ippm-multipoint-alt-mark].

This technique can be applied to point-to-point flows but also to
multipoint.to-multipoint flows (see [I-D.ietf-ippm-alt-mark] and
[I-D.fioccola-ippm-multipoint-alt-mark]). The Alternate Marking
method creates batches of packets by alternating the value of 1 or 2
bits of the packet header. These batches of packets are
unambiguously recognized over the network and the comparison of
packet counters permits the packet loss calculation. The same idea
can be applied for delay measurement by selecting special packets
with a marking bit dedicated for delay measurements. This method
needs two counters each marking period for each flow under monitor.
For this reason by considering n measurement points and n monitored
flows, the order of magnitude of the packet counters for each time
interval is n*n*2 (1 per color).

Multipoint Alternate Marking, described in
[I-D.fioccola-ippm-multipoint-alt-mark], aims to reduce this value
and makes the performance monitoring more flexible in case a detailed
analysis is not needed.

It is possible to monitor a Multipoint Network without examining in
depth by using the Network Clustering (subnetworks that are portions
of the entire network that preserve the same property of the entire
network). So in case there is packet loss or the delay is too high
the filtering criteria could be specified more in order to perform a
per flow detailed analysis, as described in [I-D.ietf-ippm-alt-mark].

An application of the multipoint performance monitoring can be done
by using FSM (each state is a composition of clusters) and feedback
mechanism where the SDN Controller is the brain of the network and
can manage flow control to the switches and routers and, in the same
way, can calibrate the performance measurements depending on the
necessity.

5. YANG for finite state machine (FSM)

This model defines a list of states and transitions to describe a
generic finite state machine (FSM). The related code and tree are
shown in the Appendix.

 <current-state>: it defines the current state of the FSM.
 <states>: this element defines the FSM as follows.
         <state>: this list defines all the FSM states.
                 <id>: this leaf attribute of <state> defines the

identifier of the state
<name>: this leaf attribute of <state> defines the
name of the state
<description>: this leaf is a "string" describing the
state
<transitions>: this attribute defines a list of
transitions to other states in the FSM.
      <name>: this attribute defines the name of a
      transition
      <type>: this attribute defines the type of the
      transition from a pool of possible transition
      types predefined inside the YANG model.
      Together with the <name> attribute, it
      uniquely identifies the transition.
      <description>: this optional attribute is a
      "string" describing the transition
      <filters>: this leaf is a list of input
      parameters related to the transition. This
      attribute enables to further express a
      transition: as an example, if a transition can
      be triggered by a parameter (e.g., a monitored
      performance parameter) exceeding a threshold
      (as in Sec. 5), an element of the list defines
      this threshold. Thus, if the parameter is
      outside the threshold, the transition is
      taken, otherwise not.
            <filter>: this leaf of <filters> defines
            a filter parameter.
            <filter-id>: this leaf of <filters>
            define the identifier number associated
            with the <filter> attribute.
      <actions>: this attribute defines a list of
      actions to take during the transition.
            <action>: this attribute is the list of
            actions
                  <id>: this leaf of <action>
                  defines the identifier number of
                  an action.
                  <type>: this leaf of <action>
                  defines the type of an action.
                  <simple>: this leaf defines
                  (differently from <conditional>
                  detailed below) an action that
                  has to be directly executed.
                        <execute>: this attribute
                        recalls an RPC encapsulating
                        the effective task (action)
                        to be executed by the

hardware. If more actions
(e.g., "A" and "B"), defined
in the <action> list, have
to be executed, these
actions can be executed
sequentially according to
the <next-action> attribute
detailed below. Thus, by
referring to the tree of the
Appendix, when an action
("A") is executed, the
<next-action> attribute will
bring to another action
("B"). If more actions have
to be executed in parallel
(e.g., "A" & "B"), not
sequentially, an element of
the <action> list should be
defined to express an action
(e.g., "A&B") consisting of
more actions to be executed
in parallel.
<next-action>: this
attribute defines the
identification number of a
next action that has to be
taken. The <next-action>
can assume a NULL value.
<conditional>: this leaf enables a
check ("true" or "false") to be
verified before executing the
action. Based on the check, the
proper attributes <execute> and
<next-operation> are considered.
<statement>: this leaf
of <conditional> defines
the condition to be
verified before executing
the action.
<true>: this leaf of
<conditional> defines a
result of the check
associated to
<statement>. Proper
<execute> and
<next-operation>
attributes are associated
with this result of the

check.
<false>: this leaf of
<conditional> defines a
result of the check
associated to
<statement>. Proper
<execute> and
<next-operation>
attributes are associated
with this result of the
check.

<next-state>: this attribute defines
the next state of FSM when an action is
executed.

6. **Implementation of the pre-programming resiliency schemes in EONs**

   These presented model can be used to enable a centralized network
   controller, managed by a network operator, to instruct data plane
   hardware on its reconfiguration if some events, such as a failure or
   physical layer degradation, occur. As an example, an optical signal
   impacted by a soft failure (i.e., a physical layer degradation
   inducing a pre forward error correction bit error rate increase -
   pre-FEC) can be maintained by adapting the FEC of the signal itself.
   This action to be taken and, more in general operations to be
   executed depending on critical events, can be (re-) programmed on the
   transponder by (re-) sending a NETCONF <edit-config> message to the
   device controller including a FSM defined by the YANG model. Such a
   system has the main goal to speed up the reaction of the network to
   certain events/faults and to alleviate the workload of the
   centralized controller. The speed up derives from the fact that the
   centralized controller is able to pre-compute and pre-configure on
   the network devices the actions to take when an event occurs taking
   into account a global view and knowledge of the network. In this
   way, the device is already aware of the actions to be locally applied
   to reconfigure a connection, avoiding to inform the controller and to
   wait for the response indicating what to do. Consequently, part of
   the workload is also removed from the centralized controller. When
   the reaction is successfully completed in the data plane, the
   centralized controller can be notified about the faults and the taken
   action. A flexible transponder supporting two FEC types, 7% and 20%,
   is considered. A two-states FSM is also assumed. The states have
   <name> attribute set to "Steady" and "Fec-Baud-Adapt", respectively.
   In the "Steady" state, the signal is in a healthy condition, adopting
   a 7% FEC, with a pre-FEC BER below an assigned threshold of 9 x 10-4.
   A transition from this state can be triggered by the event with
   <name>=BER_CHANGE and <filter-type>=9 x 10-4, thus expressing a
   change of the pre-FEC BER above the threshold. In case the pre-FEC

BER exceeds 9 x 10-4 due to a soft failure, the state machine evolves
to the "Fec-Baud-Adapt" state and an adaptation to a more robust FEC
of 20% (executed by the attribute <execute>) is performed. The
system can return to the "Steady" state if the pre-FEC BER goes below
another pre-defined threshold and the FEC is reconfigured to 7%.

7. Appendix

This appendix reports the YANG models code and the related tree.

7.1. YANG model for FSM - Tree

```
module: ietf-fsm
   +--rw current-state?   leafref
   +--rw states
      +--rw state [id]
         +--rw id             state-id-type
         +--rw description?   string
         +--rw transitions
            +--rw transition [name type]
               +--rw name          string
               +--rw type          transition-type
               +--rw description?   string
               +--rw filters
               |  +--rw filter [filter-id]
               |     +--rw filter-id    uint32
               +--rw actions
                  +--rw action [id]
                     +--rw id             transition-id-type
                     +--rw type           enumeration
                     +--rw conditional
                     |  +--rw statement    string
                     |  +--rw true
                     |  |  +--rw execute
                     |  |  +--rw next-action?   transition-id-type
                     |  |  +--rw next-state?    leafref
                     |  +--rw false
                     |     +--rw execute
                     |     +--rw next-action?   transition-id-type
                     |     +--rw next-state?    leafref
                     +--rw simple
                        +--rw execute
                        +--rw next-action?   transition-id-type
                        +--rw next-state?    leafref
```

7.2. **YANG model for FSM - Code**

<CODE BEGINS> file "ietf-fsm@2016-03-15.yang"

```
 module ietf-fsm {

   namespace "http://sssup.it/fsm";

   prefix fsm;




   identity TRANSITION {

       description "Base for all types of event";

   }



   identity ON_CHANGE {

       base TRANSITION;

       description

         "The event when the database changes.";

   }



   // typedef statements



   typedef transition-type {

        description "it defines the type of transition (event)";

     type identityref {

       base TRANSITION;
```

```
      }

    }



    typedef transition-id-type {
         description "it defines the id of the transition (event)";


      type uint32;

    }



    // grouping statements

    grouping action-block {

         description "it defines the action to perform when a transition
         occurs";

      leaf id {

description "it refers to the id of the transition";
         type transition-id-type;

      }

      leaf type {

description "it defines if the action has to be simply executed or if
a conditional statement has to be checked before execution";

         type enumeration {

           enum "CONDITIONAL_OP" {
description "it defines the type CONDITIONAL OPERATION to check a
statement before execution";
}

           enum "SIMPLE_OP" {
description "it defines the type SIMPLE OPERATION: i.e., an operation
to be directly executed;
                 }
```

```
      }

      mandatory true;

    }


    grouping execution-top {

       description "it defines the execution attribute";

      anyxml execute {

        description "Represent the action to perform";

      }

      leaf next-action {

        type transition-id-type;

        description "the id of the next action to execute";


      }

    }


    container conditional {

       description "it defines the container CONDITIONAL";

      when "../type = 'CONDITIONAL_OP'";

      leaf statement {

        type string;

        mandatory true;

        description

          "The statement to be evaluated before execution.

          E.g. if a=b";
```

```
        }

        container true {

description "it is referred to the result TRUE of a conditional
statement";

            uses execution-top;

        }

        container false {

description "it is referred to the result FALSE of a conditional
statement ";

            uses execution-top;

        }

      }


      container simple {
description "Simple execution of an action without checking any
condition";

        when "../type = 'SIMPLE_OP'";


        uses execution-top;

      }

    }


    grouping action-top {

description "it defines the grouping of action";

      list action {

description "it defines the list of actions";
```

```
      key "id";

      ordered-by user;

      uses action-block;

    }

  }


  grouping on-change {

    description

      "Event occuring when a modification of one or more

       objects occurs";


    container filters {

      description

        "This container contains a list of configurable filters

         that can be applied to subscriptions.  This facilitates

         the reuse of complex filters once defined.";

      list filter {

        key "filter-id";

        description

          "A list of configurable filters that can be applied to

           subscriptions.";

        leaf filter-id {

          type uint32;

          description
```

```
          "An identifier to differentiate between filters.";

        }

      }

    }

  }


    grouping transition-top {

description "it defines the grouping transition";

      leaf name {

description "it defines the transition name";

        type string;

        mandatory true;

      }


      leaf type {

description "it defines the transition type";

        type transition-type;

        mandatory true;

      }


      leaf description {

description "it describes the transition ";

        type string;
```

```
      }


      // list of all possible events

      uses on-change {

        when "type = 'ON_CHANGE'";

      }


      container actions {

description "it defines the container action";

        uses action-top;

      }

    }


    grouping transitions-top {

description "it defines the grouping transition";

      container transitions {

description "it defines the container transitions";

        list transition {

description "it defines the list of transitions";

          key "name type";

          uses transition-top;

        }

      }
```

```
    }


    // data definition statements


    uses transitions-top;


    // extension statements


    // feature statements


    // augment statements


    organization
      "Scuola Superiore Sant'Anna Network and Services Laboratory";


    contact
      " Editor: Matteo Dallaglio

              <mailto:m.dallaglio@sssup.it>
      ";


    description
      "This module contains a YANG definitions of a generic finite state
      machine.";
```

```
   revision 2016-03-15 {

      description "Initial Revision.";

      reference

        "RFC xxxx:";

   }



   // identity statements



   // typedef statements



   typedef state-id-type {

description "it defines the id type of the states";

      type uint32;

   }

   // grouping statements

   grouping state-top {

description "it defines the grouping state";

      leaf id {

description "it defines the id of a transition";

        type state-id-type;

      }
```

```
      leaf description {

description "it describes a transition";

        type string;

      }




      grouping next-state-top {

description "it defines the grouping for the next state";

        leaf next-state {

description "it defines the next state";

            type leafref {

description "it refers to its id";

              path "../../../../../../../../states/state/id";

            }

            description "Id of the next state";

          }

      }


    uses transitions-top {

      augment "transitions/transition/actions/action/conditional/true" {
```

```
          uses next-state-top;


      }



    augment "transitions/transition/actions/action/conditional/false" {

        uses next-state-top;

      }

    augment "transitions/transition/actions/action/simple" {

        //uses next-state-top;

        leaf next-state {

description "it defines the next state";


          type leafref {

 description "it refers to its id";
             path "../../../../../../../states/state/id";

          }

          description "Id of the next state";

        }

      }

    }



    }
```

```
   grouping states-top {

description "it defines the grouping states";

     leaf current-state {

description "it defines the current state";

       type leafref {
description "it refers to its id";
         path "../states/state/id";


       }

     }



     container states {
description "it defines the container states";

       list state {

   description "it defines the list of states";

         key "id";

         uses state-top;

       }

     }

   }



   // data definition statements
```

```
   uses states-top;


   // extension statements


   // feature statements


   // augment statements.


   // rpc statements


 }//module fsm
```


<CODE ENDS>

7.3. **Example of values for the YANG model**

FIELD NAME	YANG DATA TYPE	VALUE
Current State	leafref	"an existing state id in the FSM"
State		
id	uint32	1
name	string	Steady
description	string	"whatever string"
transition		
name	string	"whatever string"
type	enum	BER_CHANGE
description	string	"whatever string"
filter		
filter-id	uint32	2
filter-type	anyxml or xpath	BER>0.0009
action		
id	uint32	3
type	enum	SIMPLE
statement	string	"whatever string"
execute	anyxml	"this recalls an RPC where the FEC value is expressed"
next-operation	uint32	NULL
next-state	leafref	"an existing state id in the FSM"

8. Acknowledgements

9. Other Contributors

Matteo Dallaglio (Scuola Superiore Sant'Anna), Andrea Di Giglio
(Telecom Italia), Giacomo Bernini (Nextworks).

10. Security Considerations

   TBD

11. IANA Considerations

   TBD

12. References

12.1. Normative References

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

   [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>.

   [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>.

   [RFC7491] King, D. and A. Farrel, "A PCE-Based Architecture for
              Application-Based Network Operations", RFC 7491,
              DOI 10.17487/RFC7491, March 2015,
              <https://www.rfc-editor.org/info/rfc7491>.

12.2. Informative References

   [I-D.brockners-inband-oam-requirements]
              Brockners, F., Bhandari, S., Dara, S., Pignataro, C.,
              Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi,
              T., <>, P., and r. remy@barefootnetworks.com,
              "Requirements for In-situ OAM", draft-brockners-inband-
              oam-requirements-03 (work in progress), March 2017.

   [I-D.fioccola-ippm-multipoint-alt-mark]
              Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto,
              "Multipoint Alternate Marking method for passive and
              hybrid performance monitoring", draft-fioccola-ippm-
              multipoint-alt-mark-04 (work in progress), June 2018.

   [I-D.ietf-i2rs-yang-network-topo]
              Clemm, A., Medved, J., Varga, R., Bahadur, N.,
              Ananthakrishnan, H., and X. Liu, "A Data Model for Network
              Topologies", draft-ietf-i2rs-yang-network-topo-20 (work in
              progress), December 2017.

   [I-D.ietf-ippm-alt-mark]
              Fioccola, G., Capello, A., Cociglio, M., Castaldelli, L.,
              Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi,
              "Alternate Marking method for passive and hybrid
              performance monitoring", draft-ietf-ippm-alt-mark-14 (work
              in progress), December 2017.

   [I-D.song-opsawg-dnp4iq]
              Song, H. and J. Gong, "Requirements for Interactive Query
              with Dynamic Network Probes", draft-song-opsawg-dnp4iq-01
              (work in progress), June 2017.

   [I-D.vergara-ccamp-flexigrid-yang]
              Madrid, U., Perdices, D., Lopezalvarez, V., Dios, O.,
              King, D., Lee, Y., and G. Galimberti, "YANG data model for
              Flexi-Grid Optical Networks", draft-vergara-ccamp-
              flexigrid-yang-06 (work in progress), January 2018.

   [I-D.zhang-ccamp-l1-topo-yang]
              zhenghaomian@huawei.com, z., Fan, Z., Sharma, A., and X.
              Liu, "A YANG Data Model for Optical Transport Network
              Topology", draft-zhang-ccamp-l1-topo-yang-07 (work in
              progress), April 2017.

Authors' Addresses

   Nicola Sambo
   Scuola Superiore Sant'Anna
   Via Moruzzi 1
   Pisa 56124
   Italy

   Email: nicola.sambo@sssup.it

   Piero Castoldi
   Scuola Superiore Sant'Anna
   Via Moruzzi 1
   Pisa 56124
   Italy

   Email: piero.castoldi@sssup.it

Giuseppe Fioccola
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: giuseppe.fioccola@telecomitalia.it

Filippo Cugini
CNIT
Via Moruzzi 1
Pisa 56124
Italy

Email: filippo.cugini@cnit.it

Haoyu Song
Huawei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: haoyu.song@huawei.com

Tianran Zhou
Huawei
156 Beiqing Road
Beijing 100095
China

Email: zhoutianran@huawei.com