YANG Models for Vendor-Neutral Optical Networks, Reconfigurable through State Machine

Matteo Dallaglio, Nicola Sambo, Filippo Cugini, and Piero Castoldi

Multi-vendor interoperability can be achieved at node and network levels by relying on standard data modeling. YANG represents an attractive data modeling solution for network component definition. This article reports on the work done on YANG models for optical networks with particular reference to flexible-grid networks.

ABSTRACT

Multi-vendor interoperability can be achieved at node and network levels by relying on standard data modeling. YANG represents an attractive data modeling solution for network component definition. This article reports on the work done on YANG models for optical networks with particular reference to flexible-grid networks. In addition to a YANG model description for link, node, and media channels, YANG for a sliceable transponder is introduced given the importance of such a data plane device for the next generation backbone. Then a contribution is provided in proposing YANG models for events and state machine to further extend and increase the programmability of networks. This latter contribution is particularly relevant in the case of faults or physical layer degradation in a network. Finally, YANG models are validated in an experimental control plane testbed.

INTRODUCTION

Recently, network operators have shown interest in the deployment of data plane hardware providing multi-vendor interoperability [1]. This way, operators can use systems of different vendors optimizing transmission performance (e.g., achievable transmission distance), network device reuse, and capital expenditure without the need for being tied to single-vendor equipment. Multi-vendor operability can be applied in two different contexts: network and node. In the former, a network composed of nodes provided by different vendors is operated under the same control system (e.g., elastic black links [1]). In the latter, a node composed of components provided by different vendors is assembled under the same control system. This has brought about the concept of *white boxes*. With respect to *black boxes* provided by a single vendor, white boxes are assembled with different vendors' components (i.e., disaggregated hardware).

To support control and management of multi-vendor networks and white boxes, standard operator-defined data models are required so that common application programming interfaces (APIs) can be adopted for controlling/managing these multi-vendor optical systems [2].

A key candidate language to describe a standard-defined data model is the emerging Yet Another Next Generation (YANG) [3–5]. Regarding flexible/elastic optical networks [6, 7], which are the focus of this article, some recent works have provided YANG models to describe basic attributes of links (e.g., identification), nodes (e.g., connectivity matrix), media channels, and transponders (e.g., supported forward error correction, FEC) [4, 8–10].

However, effort is still required to achieve detailed models of optical devices and their functionalities to increase the level of programmability of networks. As an example, some actions have to be taken on data plane devices when events such as soft failures (i.e., performance degradation implying bit error rate [BER] increase) occur; for example, transmission should be consequently switched to a more robust modulation format [11]. Typically, actions upon failure or degradations imply either manual intervention or the involvement of a centralized controller. In the latter case, when the controller is first notified of the soft-failure event, it makes a decision and configures the involved network devices accordingly (such method requires certain time). Alternatively, the device is programmed at the moment of the installation to take an action or a reconfiguration after a specific event occurs. Currently, no YANG model has been defined to allow the controller to (re)configure events, actions, and state machine or functions on a generic network device.

In this article, we first introduce the YANG modeling language and the YANG-related works done in the field of optical networks (in particular, elastic optical networks). Then a model for the sliceable transponder is detailed. Furthermore, we propose and demonstrate the enhancement of YANG to model events and functions that can be executed in an ordered way through a finite state machine (FSM). The latter models enable a remote controller (on behalf of a network operator) to instruct a device controller about critical events and actions to be taken if these events occur. The actions to be taken and the critical events can be re-programmed on the device by simply sending a new message configuration on the device local controller with the new information.

YANG LANGUAGE

YANG [3] is a data modeling language standardized by the Internet Engineering Task Force (IETF). It has been developed in the context of NETCONF [12], a protocol standardized as

Digital Object Identifier:
10.1109/MCOM.2017.1600733

Matteo Dallaglio, Nicola Sambo, and Piero Castoldi are with Scuola Superiore Sant'Anna; Filippo Cugini is with CNIT.

an answer to specific requirements of the IETF [5]: developing standards for network configuration and management, and using XML for data encoding. Thus, NETCONF is a protocol for the configuration and management of network devices that operates on data encoded in XML. YANG has been developed and standardized as a language to model data into NETCONF messages. In particular, a YANG module can be translated into an XML representation called YIN. For this reason, commonly used XML tools can be adopted to process YANG data models, making YANG suitable for NETCONF. Thus, one of the main advantages of YANG is the XML representation, which makes YANG also adoptable by other protocols (e.g., RESTCONF) besides NETCONF. As stated in the introduction, in the last years the interest of operators in YANG has grown because of the possibility to standardize common models for configuration and management data in a vendor-neutral way. However, such models should be the synthesis of a tradeoff between different vendors. This could represent a key limitation of YANG and NETCONF, since it may result in complex common models and in a time-consuming standardization process. In this context, YANG also supports "deviations" from the common model to enable a vendor to adopt small variations with respect to the original model. YANG can be hierarchically represented in a tree structure with a root and leaves.

Figure 1 shows an example of a generic YANG model and the resulting tree organization: root and leaves have names, data types, data values, and child leaves. For example, YANG defines data types as 16-bit unsigned integers (as "data-1," line 14 in Fig. 1a) or 64-bit signed decimal (as "data-2," line 18), and others [3]. New data types can be also defined. In Fig. 1a, "data-3" is of "NEW-TYPE" type. In the example, the new type can assume just three values (lines 5–12). YANG also includes the definition of lists. The "key" of a list is used to specify one or more leaves in a list that will uniquely identify an element (data instance) of the list. The example in Fig. 1a shows the model "example" composed of four leaves (each leaf is defined with the syntax "leaf," e.g., line 13): "data-1," "data-2," "data-3," and a list (line 27). Each piece of this data is associated with a type. A list is initiated with the command "list" (line 27), and the data of a list can have child leaves as "leaf-data-1" and "leaf-data-2." The resulting tree, obtained with the Pyang software [9], is visualized in Fig. 1b with "example" as the root; "data-1," "data-2," "data-3," and the list as leaves; and "leaf-data-1" and "leaf-data-2" as the leaves of each element in the list.

(a)

```
1.          module example {
2.              namespace "sssup:example";
3.          prefix example ;
4.
5.          typedef NEW-TYPE{
6.              type enumeration {
7.                  enum type-one;
8.                  enum type-two;
9.                  enum type-three;
10.             }
11.         }
12.
13.         leaf data-1 {
14.             type uint16 ;
15.         }
16.
17.         leaf data-2 {
18.             type decimal64 {
19.                         fraction-digits 18;
20.             }
21.         }
22.
23.         leaf data-3 {
24.             type NEW-TYPE ;
25.         }
26.
27.         list element-of-a-list {
28.             key "leaf-data-1";
29.             leaf leaf-data-1 {
30.                 type uint16;
31.             }
32.             leaf leaf-data-2 {
33.                 type uint16;
34.             }
35.     }
36. }
```

(b)

```
module: example
    +--rw data-1?                 uint16
    +--rw data-2?                 decima164
    +--rw data-3?                 NEW-TYPE
    +--rw element-of-a-list       [leaf-data-1]
        +--rw leaf-data-1         uint16
        +--rw leaf-data-2?        uint16
```

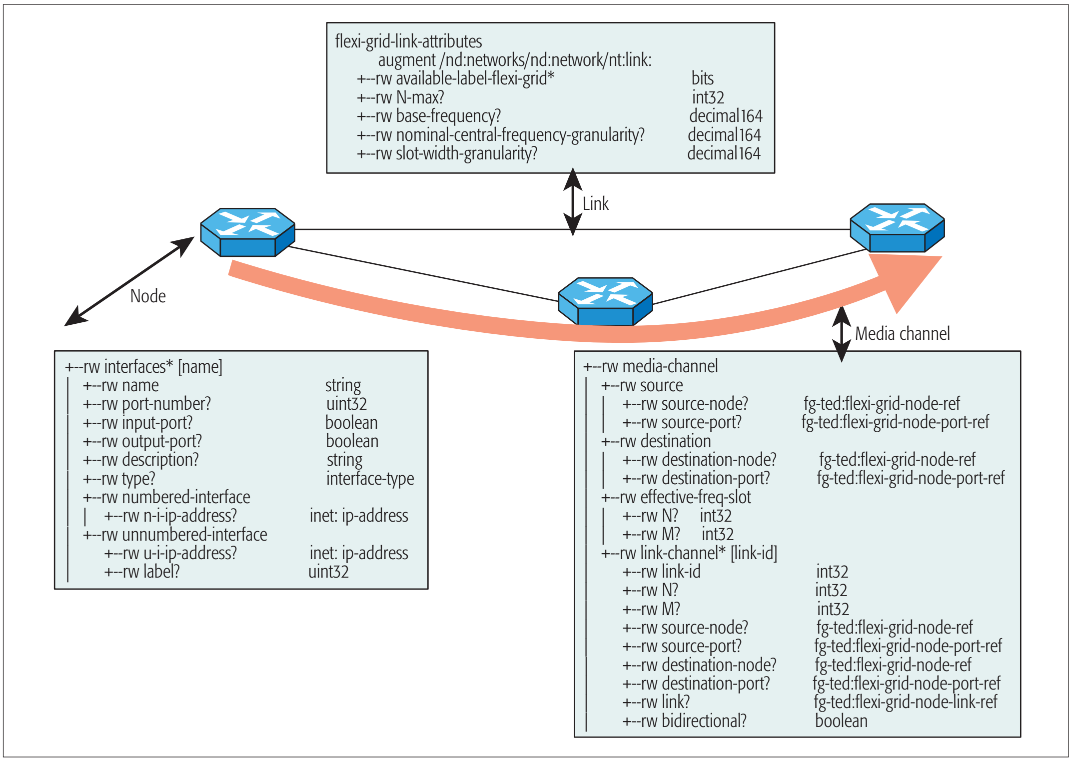
Figure 1. a) Example of YANG code; b) resulting tree.

YANG data can be of two types: *configuration* or *state*. Configuration data is explicitly set by an external entity from the system (e.g., the centralized controller). State data cannot be set by the external entity, but they can be read. State data can be used for monitoring purposes. A further layer in the hierarchy indicating the list of configuration and state data can be defined, as detailed later. YANG also supports the definition of "Notification" to model the content of NETCONF Notification messages, which indicate that certain events have been recognized (e.g., a failed link). Moreover, although YANG is mostly considered as a data modeling language, it also provides the possibility to define executable functions through remote procedure calls (RPCs) that specify the name, the input, and the output parameters of a specific function, for example, switching on (off) a device inside a node. For further information the reader is referred to [3].

YANG data can be of two types: configuration or state. Configuration data is explicitly set by an external entity from the system. State data cannot be set by the external entity but they can be read. State data can be used for monitoring purposes. A further layer in the hierarchy indicating the list of configuration and state data can be defined.

Then some considerations are here reported on the nature of YANG. First, it is a highly readable text language. This significantly simplifies management and troubleshooting operations compared to protocols relying on bit encoding, which require ad hoc software to parse encoded information. Nowadays, handling a text file instead of bit encoding does not represent a particular challenge. Moreover, in the case of bit encoding, the support of novel parameters at the data plane would imply redesigning the protocol messages' content, such as header and

Thanks to the nature of YANG, when the model changes, the YANG model can be refined without redesigning the protocol, thus providing a much more effective solution with respect to bit encoding. Such an example has to be considered relevant given the continuous evolution of the technology at the data plane.



```
flexi-grid-link-attributes
      augment /nd:networks/nd:network/nt:link:
   +--rw available-label-flexi-grid*                     bits
   +--rw N-max?                                          int32
   +--rw base-frequency?                                 decimal164
   +--rw nominal-central-frequency-granularity?          decimal164
   +--rw slot-width-granularity?                         decimal164
```

```
+--rw interfaces* [name]
|  +--rw name                            string
|  +--rw port-number?                    uint32
|  +--rw input-port?                     boolean
|  +--rw output-port?                    boolean
|  +--rw description?                    string
|  +--rw type?                           interface-type
|  +--rw numbered-interface
|  |  +--rw n-i-ip-address?            inet: ip-address
|  +--rw unnumbered-interface
|     +--rw u-i-ip-address?            inet: ip-address
|     +--rw label?                     uint32
```

```
+--rw media-channel
|  +--rw source
|  |  +--rw source-node?          fg-ted:flexi-grid-node-ref
|  |  +--rw source-port?          fg-ted:flexi-grid-node-port-ref
|  +--rw destination
|  |  +--rw destination-node?       fg-ted:flexi-grid-node-ref
|  |  +--rw destination-port?       fg-ted:flexi-grid-node-port-ref
|  +--rw effective-freq-slot
|  |  +--rw N?    int32
|  |  +--rw M?    int32
|  +--rw link-channel* [link-id]
|     +--rw link-id                 int32
|     +--rw N?                      int32
|     +--rw M?                      int32
|     +--rw source-node?            fg-ted:flexi-grid-node-ref
|     +--rw source-port?            fg-ted:flexi-grid-node-port-ref
|     +--rw destination-node?       fg-ted:flexi-grid-node-ref
|     +--rw destination-port?       fg-ted:flexi-grid-node-port-ref
|     +--rw link?                   fg-ted:flexi-grid-node-link-ref
|     +--rw bidirectional?          boolean
```

Figure 2. Portions of a YANG tree representation of flexible optical networks as proposed by [4]: node, link, and media channel.

objects. On the contrary, thanks to the nature of YANG, when the model changes, the YANG model can be refined without redesigning the protocol, thus providing a much more effective solution with respect to bit encoding. Such an example has to be considered relevant given the continuous evolution of the technology at the data plane.

In the context of optical networks, several standardization bodies and working groups (e.g., IETF, OpenConfig, OpenROADM) have released YANG models. As an example, the IETF draft in [13] defines a YANG model for representing, retrieving, and manipulating traffic engineering (TE) topologies supporting optical switching nodes. OpenROADM has recently defined YANG models focused on reconfigurable optical add/drop multiplexer (ROADM) disaggregation. These models describe how different pluggable devices for optical networks (e.g., amplifiers, transponders) can be interconnected. However, more details on the transponder parameters (e.g., chromatic dispersion, polarization mode dispersion, analog bandwidth) could be provided. OpenConfig aims to provide a set of vendor-neutral data models based on network operator requirements. In particular, OpenConfig released preliminary models on optical amplifiers, ROADMs, and transponders. The OpenConfig model does not consider disaggregation as OpenROADM does, while the transponder model is more accurate with respect to OpenROADM but still lacks some parameters (e.g., sampling rate, analog bandwidth) and does not define any Notification that can be very relevant for monitoring purposes [8].

In the next section, YANG models for flexible (or elastic) optical networks are introduced.

YANG MODEL FOR ELASTIC OPTICAL NETWORKS

Elastic optical networks (EONs) are circuit-switched optical networks equipped with flexible-grid spectrum selective switches (SSSs) [11]. SSSs enable switching of the configurable portion of the bandwidth, depending on the bandwidth required by the circuit or *media channel* (e.g., by fixing the modulation format, a high-rate connection requires more bandwidth than a lower-rate connection). The media channel is defined as a specific portion of the optical spectrum along an optical path between a source and a destination node [11]. For EONs, International Telecommunication Union — Telecommunication Standardization Sector (ITU-T) G.694.1 states that a media channel occupies a portion of spectrum called *frequency slot*, defined by two parameters: the *central frequency* and the *width* of the occupied spectrum portion. According to this ITU-T specification, the central frequency can assume values in steps of 6.25 GHz, while the width has to be a multiple of 12.5 GHz. In [4], the authors have been focused on the representation of the flexi-grid optical layer dividing the model into two modules: one related to the TE database (TED) and the other one representing the *media channel*. The TED module defines the information required to represent nodes, links, transponders, and spectrum resources. Portions of the trees of these sub-modules are shown in Fig. 2.

The sub-module of the transponder, being more complex, is detailed in the next section. The "interfaces" leaf of the node sub-module is a list containing all the interfaces in the node. Each element of this leaf has several sub-leaves defining attributes of the considered interface (or port), such as the name, the number, two Boolean variables indicating if it is an input or an output port, and the IP address if present. The model also includes the "connectivity matrix" (not shown in the figure): a list of connected input/output ports in the node. Additional information may be added. This model can be further augmented by including information on the add/drop part of the node, in particular, to define the reachability of an add port (or a drop port) to an output interface (or an input interface). More information on add/drop is included in the YANG model in [10].

The link sub-module consists of five leaves: the availability of flex-grid technology for that link, the maximum value *N* of slices supported by that link (i.e., slices of 12.5 GHz), the nominal central frequency for the link, the spacing among channels' central frequency (i.e., 6.25 GHz), and the slot width granularity (i.e., 12.5 GHz). The media channel sub-module consists of four main leaves: the source and destination nodes of the media channel, the frequency slot, and a list of traversed links. Both source and destination nodes include two leaves: one defining a reference to the module of the node (i.e., the tree of Fig. 2) and the other one related to the used interface (port) in such a node. This model can be further augmented including a reference to the transponder used by the media channel, the used add/drop port, and also information on the adopted transmission technique (e.g., Nyquist wavelength-division multiplexing, NWDM).

SLICEABLE TRANSPONDER

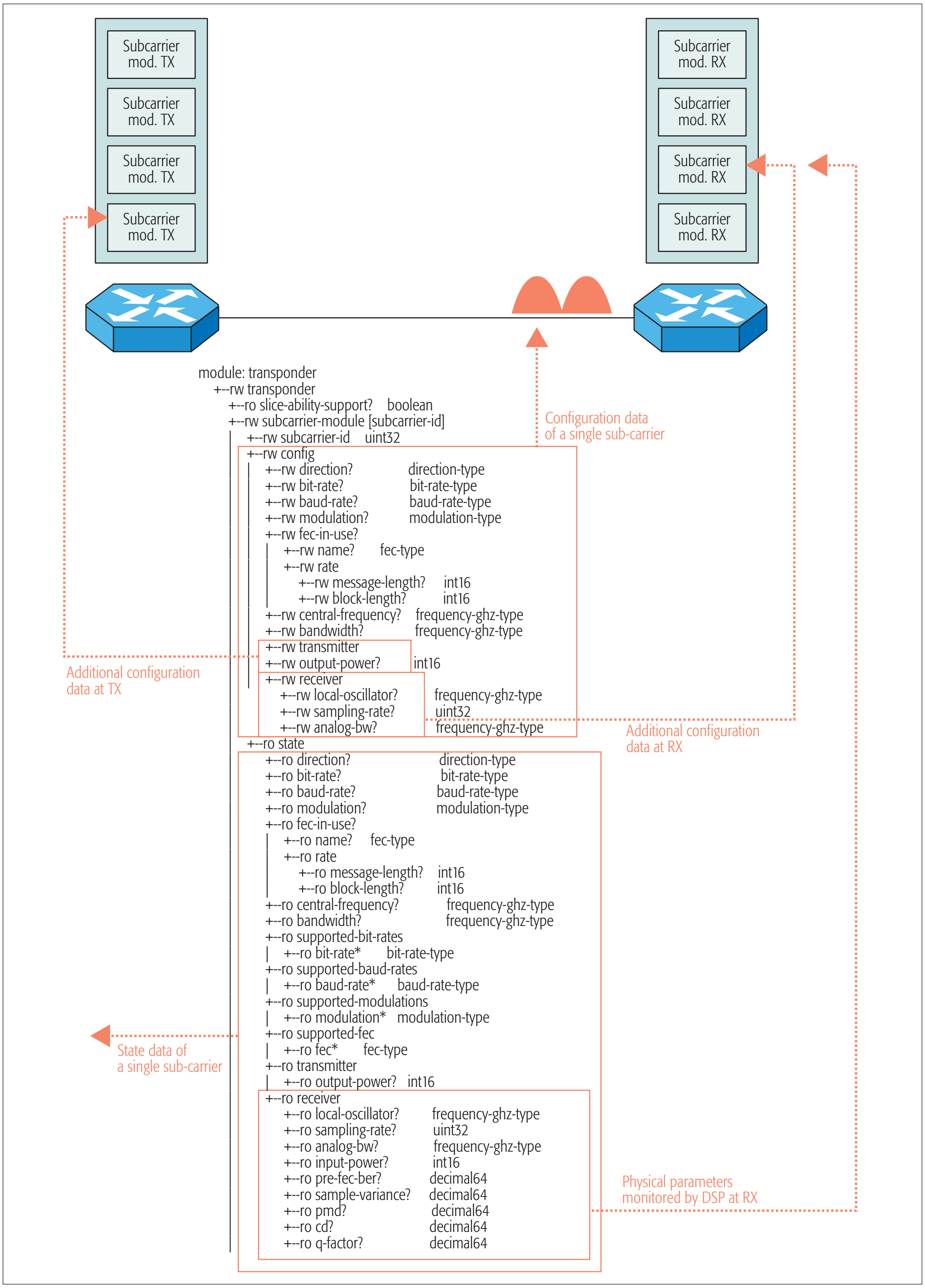
A sliceable transponder is a transponder generating multiple independent optical flows that can be directed toward different destinations [11]. A reference architecture agreed on among several vendors and operators has been proposed in [11]. In this article, we mainly refer to the transponder model in [9], which reports a comprehensive set of physical parameters with a particular reference to state data that can be used for monitoring purposes. In [9], the authors enhanced the YANG model for the sliceable transponder by leveraging on the one presented in [4]. In particular, more physical data has been included in the YANG model (e.g., baud rate, output power at the transmitter side, the local oscillator and the analog bandwidth at the receiver, monitoring parameters that are detailed in this section, and a reference to the media channels using the transponder). Moreover, a classification on the configurable and state data is provided. This YANG model reflects the transponder architecture of [11]. The transponder is composed of a set of subcarrier modules. Each subcarrier module is devoted to generating (at the transmitter side) or detecting (at the receiver side) an optical subcarrier. Similarly, the YANG model is organized per subcarrier module. The related tree is shown in Fig. 3. First, a Boolean data indicates if slice-ability is supported or not. Then a list of subcarriers' sub-modules is modeled. As configuration data, different data are present if the "direction" is in transmission or detection (e.g., local oscillator configuration if the module is in detection). Other data has to be specified in both transmission and detection: for example, baud rate, bit rate, modulation format, FEC. Note that we defined the type "frequency-ghz-type" to discern between the central frequency of a subcarrier and that of a media channel. Indeed, while the central frequency of a media channel has to follow ITU-T specifications in steps of 6.25 GHz, and thus can be expressed as just an integer number, the central frequency of a subcarrier of a media channel composed of several subcarriers does not necessary follow a grid [11]. Thus, the central frequency of a subcarrier can be any number. For this reason, we defined the type "frequency-ghz-type" to express the frequency value in "GHz." Regarding state data, first, configuration data is replicated into state data to enable an operator to verify ("read") the actual configuration of the transponder. Then other data is included in the model, mainly related to the monitoring capabilities of coherent detection. Indeed, thanks to the digital signal processing (DSP) at the receiver, it is possible to monitor end-to-end parameters associated with each subcarrier [11]. As an example, monitored parameters can be pre-FEC BER, Q-factor, chromatic dispersion (CD), and polarization mode dispersion (PMD), all expressed as decimal64. Other leaves of the subcarrier module comprise (not shown in the figure) the identification of the node and of the add/drop module, and a list of media channels that are using such a transponder. Finally, different from the representation in [4], the "transmission scheme" is included to identify the adopted transmission technique. For that, a new type is defined including NWDM, orthogonal frequency-division multiplexing (OFDM), and others. The full code of this model can be retrieved from [14].

EVENTS AND STATE MACHINE

A sliceable transponder can be reconfigured when some events occur [15]: for example, degradations of the physical layer due to aging may imply an increase of the pre-FEC BER. Such an event can be overcome by making the transmission more robust (e.g., by changing the modulation format or the FEC). This section is devoted to model events, actions, and FSMs. Such models are proposed to enable a remote controller (on behalf of a network operator) to instruct a device controller about critical events and actions to be taken if this event occurs. The actions to be taken and the critical events can be reprogrammed on the device by simply resending a new message configuration (e.g., through the NETCONF protocol, as detailed in the next section) on the device controller with the new information. Such a system has the prospect to speed up the reaction of the network to certain events/faults and to alleviate, in a standard way, the workload of the centralized controller. The speedup derives from the fact that the centralized controller is able to pre-configure, on the network devices, the actions to take when an event occurs. In this way, the device already knows what to do and can immediately react, avoiding informing the controller and waiting for the response indicating what to do. Consequently, part of the workload is also removed from the centralized controller, which can instruct the device once, transferring to it some intelligence to make decisions autonomously. When the reaction is successfully completed in the data plane, the centralized controller can be notified about the faults and the action taken.

In the context of optical networks, several standardization bodies and working groups (e.g., IETF, OpenConfig, OpenROADM) have released YANG models..

The use of YANG and, in particular, finding common models for events and transceiver actions/functions can be considered of relevance because of two main trends: network operators looking for common vendor-neutral solutions; and developing transponders supporting multiple transmission parameters and monitoring capabilities.



```
module: transponder
  +--rw transponder
     +--ro slice-ability-support?   boolean
     +--rw subcarrier-module [subcarrier-id]
     |  +--rw subcarrier-id    uint32
        +--rw config
        |  +--rw direction?            direction-type
        |  +--rw bit-rate?             bit-rate-type
        |  +--rw baud-rate?            baud-rate-type
        |  +--rw modulation?           modulation-type
        |  +--rw fec-in-use?
        |  |  +--rw name?      fec-type
        |  |  +--rw rate
        |  |     +--rw message-length?   int16
        |  |     +--rw block-length?     int16
        |  +--rw central-frequency?   frequency-ghz-type
        |  +--rw bandwidth?           frequency-ghz-type
        |  +--rw transmitter
        |  +--rw output-power?        int16
        |  +--rw receiver
        |     +--rw local-oscillator?     frequency-ghz-type
        |     +--rw sampling-rate?        uint32
        |     +--rw analog-bw?            frequency-ghz-type
        +--ro state
           +--ro direction?              direction-type
           +--ro bit-rate?               bit-rate-type
           +--ro baud-rate?              baud-rate-type
           +--ro modulation?             modulation-type
           +--ro fec-in-use?
           |  +--ro name?     fec-type
           |  +--ro rate
           |     +--ro message-length?   int16
           |     +--ro block-length?     int16
           +--ro central-frequency?        frequency-ghz-type
           +--ro bandwidth?                frequency-ghz-type
           +--ro supported-bit-rates
           |  +--ro bit-rate*     bit-rate-type
           +--ro supported-baud-rates
           |  +--ro baud-rate*     baud-rate-type
           +--ro supported-modulations
           |  +--ro modulation*   modulation-type
           +--ro supported-fec
           |  +--ro fec*       fec-type
           +--ro transmitter
           |  +--ro output-power?   int16
           +--ro receiver
              +--ro local-oscillator?     frequency-ghz-type
              +--ro sampling-rate?        uint32
              +--ro analog-bw?            frequency-ghz-type
              +--ro input-power?          int16
              +--ro pre-fec-ber?          decimal64
              +--ro sample-variance?      decimal64
              +--ro pmd?                  decimal64
              +--ro cd?                   decimal64
              +--ro q-factor?             decimal64
```

Figure 3. YANG tree representation of a sliceable transponder.

The use of YANG and, in particular, finding common models for events and transceiver actions/functions can be considered relevant because of two main trends: network operators looking for common vendor-neutral solutions; and developing transponders supporting multiple transmission parameters (e.g., bit rate, coding, modulation format, baud rate) and monitoring capabilities. Moreover, several activities of operators and vendors are evaluating the reduction of network margins [15] (i.e., worst-case margins for aging and transmission modeling inaccuracy), for example, to decrease the number of opto-electronic converters. This will cause networks to suffer more from changes in the physical layer (e.g., due to events such as soft failures), thus increasing the needs of devices supporting transmission adaptation (e.g., to increase robustness).

The proposed YANG model, schematized with the tree diagram of Fig. 4a, describes events (e.g., soft failures) and functions (e.g., baud rate and code change) to be executed in an ordered way following an FSM. The model defines a list of events as the root of the hierarchy. An event is defined through two mandatory attributes (“name” and “type”) and an optional attribute (“description”). Together, “name” and “type” attributes uniquely identify the event. The “type” attribute takes its value from a pool of possible event types predefined inside the YANG model. Currently, we have defined some known event types such as the “ON CHANGE” event to describe the change of an attribute value. Given that the change of an attribute does not necessarily mean a particular degradation or fault, we included in the model the sub-leaf “filter,” which can be used to define a threshold to further characterize the event. For example, by referring to the “Q-factor” state data in Fig. 3, we may define an event named Q-factor change of type “ON CHANGE” and, as a filter, a threshold to indicate when the Q-factor falls below the threshold. Another leaf of the “event” is the “reaction.” In particular, for each event, the controller can configure a reaction the device should have. The “reaction” is composed of a list of “operations” to perform when the event occurs. Each operation is identified through an “id” and can be either of types “simple” or “conditional.” A “simple” operation contains the “execute” attribute that, recalling an RPC (as shown in the next section), is used to encapsulate the effective task to be executed and the “id” of the “next operation” (if any). A “conditional” operation, with respect to the “simple” one, contains in addition a “statement” attribute that can be “true” or “false” (related flow chart shown in Fig. 4b). The statement is checked at the beginning of the operation; then, depending on the outcome (true or false), only the correct operation is considered. “True” and “false” contain the “execute” and “next operation” as for the “simple” operation.

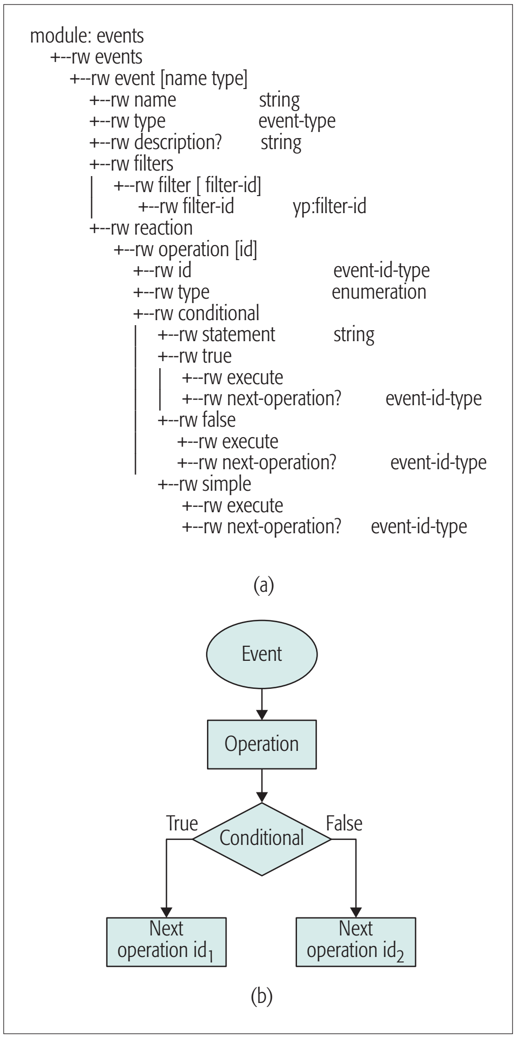
It is important to underline that this proposed model does not replace notification; indeed, the centralized controller should always be notified when an event occurs. However, in the meantime the device can already start reacting to the event. It is also important to note that reactions are not statically pre-configured; they can be revoked or reconfigured by the controller depending on the evolution of the network (e.g., depending on bandwidth availability).

It is important to underline that this proposed model does not replace notification; indeed, the centralized controller should always be notified when an event occurs. However, in the meantime the device can already start reacting to the event.

We also propose a YANG model for an FSM. Each state of the machine is based on the Event YANG model. In particular, the FSM YANG model extends the YANG model for the events by adding the state information and state transition. More precisely, the model defines a list of states that, similar to the events, are configurable by the controller. Each state has a description attribute and it is identified through an ID. Each state also includes a list of events as defined in the event model, with the additional next-state attribute, which points to the next state.

EXPERIMENTAL DEMONSTRATION OF YANG-BASED CONTROL PLANE MODELING EVENTS AND STATE MACHINE

The proposed models have been experimentally demonstrated in a testbed composed of a centralized network controller (implementing phyton) and two transponder controllers (using ConfD) at the transmitter and receiver side, respectively.



```
module: events
   +--rw events
      +--rw event [name type]
         +--rw name              string
         +--rw type              event-type
         +--rw description?      string
         +--rw filters
         |   +--rw filter [ filter-id]
         |      +--rw filter-id          yp:filter-id
         +--rw reaction
            +--rw operation [id]
               +--rw id                        event-id-type
               +--rw type                      enumeration
               +--rw conditional
               |  +--rw statement          string
               |  +--rw true
               |  |  +--rw execute
               |  |  +--rw next-operation?       event-id-type
               |  +--rw false
               |     +--rw execute
               |     +--rw next-operation?       event-id-type
               +--rw simple
                  +--rw execute
                  +--rw next-operation?     event-id-type
```

(a)

(b)

Figure 4. a) YANG tree representation of events and reactions; b) flow chart for conditional operations.

First, the transponder YANG model discussed earlier is considered, and a NETCONF message is generated to configure the following transmission parameters: 100 Gb/s net rate connection with a baud rate of 28 Gbaud, 7 percent of FEC, and polarization multiplexing quadrature phase shift keying (PM-QPSK) modulation format.

Then the configuration of events and state machine is performed as in Fig. 5a, which shows the NETCONF message exchange between the centralized controller and a transponder controller at the transmitter side. Similarly, message exchange has been performed with the controller at the receiver side. Initially, the centralized controller sends an <edit-config> message, as in [5], including the structure of the FSM and the associated events. This message enables the remote controller to instruct the device controller about FSM, critical events, and actions to be taken if these events occur. Once the device controller is instructed about FMS and the event, an acknowledgment message (<ok> message as in [5]) is sent to the remote centralized controller notifying that the operation has been concluded. The actions to be taken and the critical events can be reprogrammed on the device by simply sending a new message configuration to the device controller

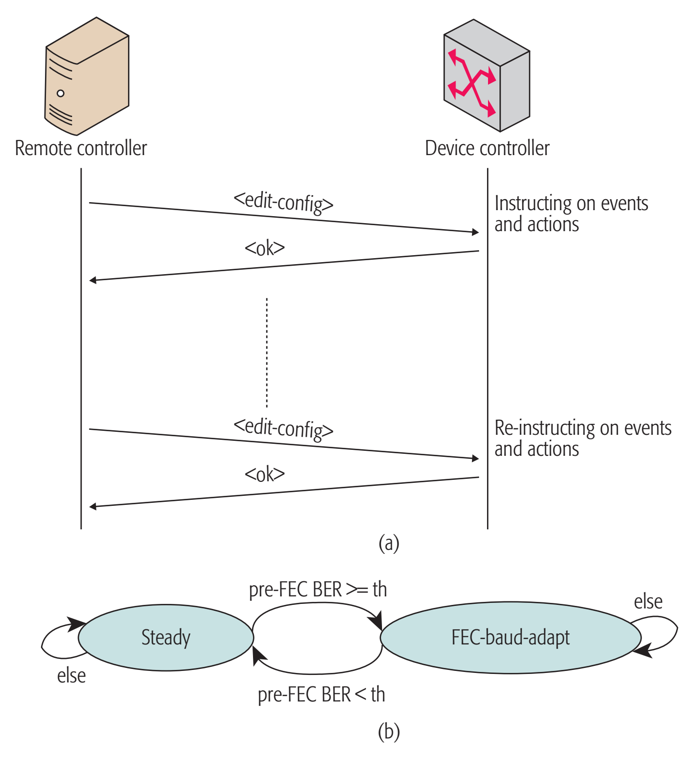


Figure 5. a) NETCONF message exchange in the testbed; b) implemented finite state machine.

with the new information. The experiment consists of configuring the FSM depicted in Fig. 5b, which is composed of two states: “Steady” and “Fec-Baud-Adapt.” In the Steady state, the connection is in a healthy condition with a pre-FEC BER below the assigned threshold of 9×10^{-4}. If the pre-FEC BER exceeds the threshold, the state machine evolves to the Fec-Baud-Adapt state, where an adaptation to a more robust FEC (20 percent) and a baud rate change (to 31 Gbaud) are performed. Note that the centralized controller is aware of spectrum occupation. The receiver controller detecting the failure sends a notification of the event to the transmitter controller through the supervisory channel. This way, the transmitter controller reconfigures the transmission parameters (FEC) based on the event and the instructions in its FSM. From the FEC-Baud-Adapt state, if the pre-FEC BER returns below the threshold, the state machine moves back to the Steady state, readjusting the baud rate and the FEC to the initially configured values.

Figure 6a shows a portion of the message sent by the controller to the transponder to configure the FSM previously described. In particular, the Steady state with id 1 and Fec-Baud-Adapt with id 2 can be identified. The Steady state is the starting point as indicated by the current-state attribute. It responds to the “ON CHANGE” event, more precisely only when the pre-FEC BER changes to a value higher than 9×10^{-4}. The associated reaction to the event is composed of a single operation (“execute”). As stated in the previous section, the “execute” command recalls an RPC (Fig. 6a) consisting of changing the baud rate and the FEC. After the execution, the current state becomes the state with id 2 (Fec-Baud-Adapt) as indicated by the next-state attribute. The “Fec-Baud-Adapt” state also responds to the “ON CHANGE” EVENT, but in this case only when the pre-FEC BER goes below the threshold. Similar to the Steady state, a single operation is executed in reaction. In this case, the same RPC is recalled with different values: the FEC and the baud rate are restored to the initial values.

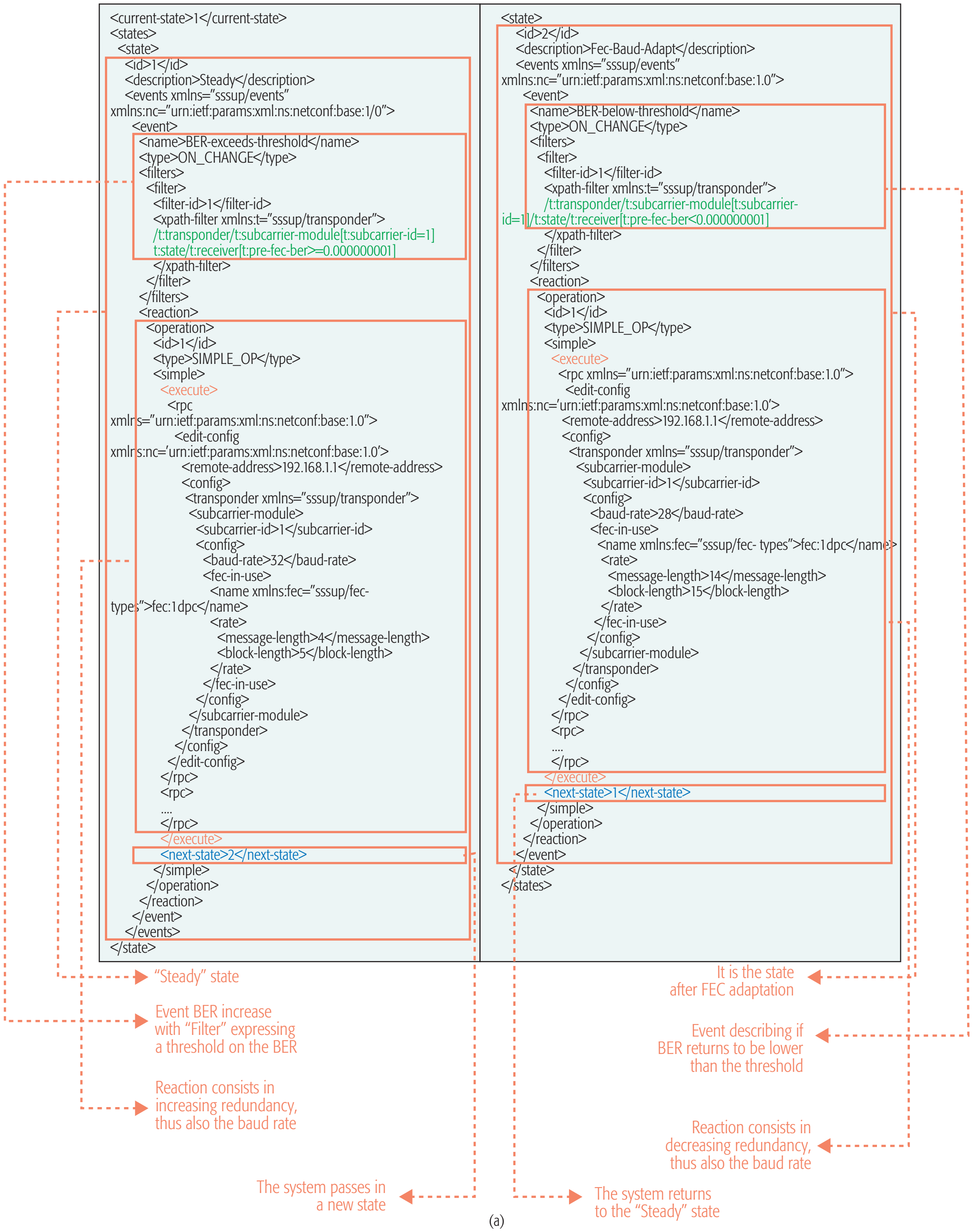
This way, the transponder device controller is successfully configured and instructed about the actions to perform when specific events occur. In the case of pre-FEC BER increase (or decrease), the transponder is able to automatically reconfigure itself without requesting the centralized controller and then waiting for its response on the actions to perform (only a notification message is generated).

Finally, we exploited simulations on a Spanish backbone (the same topology used in [15]) to identify the average number of 100 Gb/s PM-QPSK lightpaths affected by a soft failure. Results are shown in Fig. 6b. We generated an optical signal-to-noise ratio (OSNR) penalty spanning from 1 to 3 dB on random links. A lightpath is considered affected by the soft failure if the OSNR penalty causes a pre-FEC BER increase above the threshold of 10^{-3}; otherwise, the light path is assumed to be robust and can continue its normal transmission. The number of affected lightpaths increases with OSNR penalty since a higher penalty causes a higher pre-FEC BER increase. In the traditional case, the centralized controller has to receive notifications about the failure and the affected lightpaths, take a decision per lightpath (e.g., FEC adaptation), and send a message to reconfigure the involved devices. Thus, for high OSNR penalty, the centralized controller is also more loaded and reconfiguration at the data plane can suffer from delay. For example, in the case of 2 dB OSNR penalty, an average number of 16 lightpaths is affected. Conversely, a system exploiting the proposed YANG model for FSM is more scalable since the centralized controller is only notified upon failure.

CONCLUSIONS

In this article, the YANG modeling language has been described and enhanced to enable effective multi-vendor interoperability operations at both the network and node (i.e., white box) levels. Indeed, by standardizing a common language for network and node parameters, a controller can control and manage devices provided by different vendors, positively impacting the overall capital expenditure without being tied to single vendor’s equipment. Specific enhancement has been introduced to also enable the YANG language to describe events and finite state machines, thus describing the set of actions to be performed at the node or device level without centralized controller intervention.

The defined YANG models for transponder, events, and finite state machines have been used in a control plane testbed to successfully configure, in a vendor-independent way, both transmission parameters and the actions to perform upon the occurrence of specific events. This way, upon pre-defined events at the physical layer (e.g., BER increase), the transponder is able to autonomously react without requiring time-consuming interaction with the centralized controller.



```
<current-state>1</current-state>
<states>
 <state>
  <id>1</id>
  <description>Steady</description>
  <events xmlns="sssup/events"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1/0">
   <event>
    <name>BER-exceeds-threshold</name>
    <type>ON_CHANGE</type>
    <filters>
     <filter>
      <filter-id>1</filter-id>
      <xpath-filter xmlns:t="sssup/transponder">
      /t:transponder/t:subcarrier-module[t:subcarrier-id=1]
      t:state/t:receiver[t:pre-fec-ber>=0.000000001]
      </xpath-filter>
     </filter>
    </filters>
    <reaction>
     <operation>
      <id>1</id>
      <type>SIMPLE_OP</type>
      <simple>
       <execute>
        <rpc
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
         <edit-config
xmlns:nc='urn:ietf:params:xml:ns:netconf:base:1.0'>
          <remote-address>192.168.1.1</remote-address>
          <config>
           <transponder xmlns="sssup/transponder">
            <subcarrier-module>
             <subcarrier-id>1</subcarrier-id>
             <config>
              <baud-rate>32</baud-rate>
              <fec-in-use>
               <name xmlns:fec="sssup/fec-
types">fec:1dpc</name>
               <rate>
                <message-length>4</message-length>
                <block-length>5</block-length>
               </rate>
              </fec-in-use>
             </config>
            </subcarrier-module>
           </transponder>
          </config>
         </edit-config>
        </rpc>
        <rpc>
        ....
        </rpc>
       </execute>
       <next-state>2</next-state>
      </simple>
     </operation>
    </reaction>
   </event>
  </events>
</state>
```

```
<state>
  <id>2</id>
  <description>Fec-Baud-Adapt</description>
  <events xmlns="sssup/events"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <event>
    <name>BER-below-threshold</name>
    <type>ON_CHANGE</type>
    <filters>
     <filter>
      <filter-id>1</filter-id>
      <xpath-filter xmlns:t="sssup/transponder">
      /t:transponder/t:subcarrier-module[t:subcarrier-
id=1]/t:state/t:receiver[t:pre-fec-ber<0.000000001]
      </xpath-filter>
     </filter>
    </filters>
    <reaction>
     <operation>
      <id>1</id>
      <type>SIMPLE_OP</type>
      <simple>
       <execute>
        <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
         <edit-config
xmlns:nc='urn:ietf:params:xml:ns:netconf:base:1.0'>
          <remote-address>192.168.1.1</remote-address>
          <config>
           <transponder xmlns="sssup/transponder">
            <subcarrier-module>
             <subcarrier-id>1</subcarrier-id>
             <config>
              <baud-rate>28</baud-rate>
              <fec-in-use>
               <name xmlns:fec="sssup/fec- types">fec:1dpc</name>
               <rate>
                <message-length>14</message-length>
                <block-length>15</block-length>
               </rate>
              </fec-in-use>
             </config>
            </subcarrier-module>
           </transponder>
          </config>
         </edit-config>
        </rpc>
        <rpc>
        ....
        </rpc>
       </execute>
       <next-state>1</next-state>
      </simple>
     </operation>
    </reaction>
   </event>
  </state>
 </states>
```

"Steady" state

Event BER increase with "Filter" expressing a threshold on the BER

Reaction consists in increasing redundancy, thus also the baud rate

The system passes in a new state

It is the state after FEC adaptation

Event describing if BER returns to be lower than the threshold

Reaction consists in decreasing redundancy, thus also the baud rate

The system returns to the "Steady" state

(a)

OSNR penalty [dB]	Average number of affected lightpaths	Operations at the controller upon failure (traditional approach)	Operations at the controller upon failure if FSM YANG is exploited
1	9.14	1) Receiving notifications about failure detection and involved lightpaths 2) Computation of recovery strategy per lightpath (e.g., FEC adaptation) 3) Sending messages for reconfigurations	1) Receiving notifications about failure detection, involved/ recovered lightpaths (e.g., FEC adaptation)
2	16.42		
3	22.52		

(b)

Figure 6. a) Capture of the control plane message instructing the device controller about the state machine; b) lightpaths involved in a failure and related operations at the centralized controller.

In the future, efforts in data modeling among vendors and operators will follow up to find common standard solutions. Moreover, functional models, such as finite state machines, can be enriched by adding new constructs besides "simple" and "conditional" operations (e.g., error checking, and loops such as "for" and "while").

In the future, efforts in data modeling among vendors and operators will follow up to find common standard solutions. Moreover, functional models, such as finite state machine, can be enriched by adding new constructs besides "simple" and "conditional" operations.

ACKNOWLEDGMENT

This work was supported by the EC through the Horizon 2020 ORCHESTRA project (grant agreement 645360).

REFERENCES

[1] M. Gunkel *et al.*, "Vendor-Interoperable Elastic Optical Interfaces: Standards, Experiments, and Challenges," *IEEE/OSA J. Optical Commun. Networking*, vol. 7, no. 12, Dec. 2015, pp. B184–93.

[2] A. Shaikh *et al.*, "Vendor-Neutral Network Representations for Transport SDN" *Proc. 2016 OFC*, Anaheim, CA, Mar. 2016, pp. 1–3.

[3] M. Bjorklund, "YANG — A Data Modeling Language for the Network Configuration Protocol (NETCONF)," IETF RFC 6020, Oct. 2010.

[4] J. Vergara *et al.*, IETF draft-vergara-ccamp-flexigrid-yang-03, July 2016.

[5] J. Schonwalder, M. Bjorklund, and P. Shafer, "Network Configuration Management Using NETCONF and YANG," *IEEE Commun. Mag.*, vol. 48, no. 9, Sept 2010, pp. 166–73.

[6] M. Jinno *et al.*, "Distance-Adaptive Spectrum Resource Allocation in Spectrum-Sliced Elastic Optical Path Network," *IEEE Commun. Mag.*, Topics in Optical Communications, vol. 48, no. 8, Aug. 2010, pp. 138–45.

[7] L. Liu *et al.*, "OpenSlice: An OpenFlow-Based Control Plane for Spectrum Sliced Elastic Optical Path Networks," *Optics Express*, vol. 21, no. 4, Feb. 2013, pp. 4194–4204.

[8] https://github.com/openconfig/public/tree/master/release/models/optical-transport' accessed Dec. 2016

[9] M. Dallaglio *et al.*, "YANG Model and NETCONF Protocol for Control and Management of Elastic Optical Networks," *Proc. 2016 OFC*, Anaheim, CA, 2016, pp. 1–3.

[10] https://github.com/OpenROADM/OpenROADM_MSA_Public/tree/master/Version percent201.2, accessed Oct. 2016.

[11] N. Sambo *et al.*, "Next Generation Sliceable Bandwidth Variable Transponders," *IEEE Commun. Mag.*, vol. 53, no. 2, Feb. 2015, pp. 163–71.

[12] R. Enns *et al.*, "Network Configuration Protocol (NETCONF)," IETF RFC 6241, June 2011.

[13] X. Liu *et al.*, IETF draft-ietf-teas-YANG-te-topo-06, Oct. 2016.

[14] https://github.com/mattedallo/sssa/blob/master/yang-models/transponder.yang, accessed Mar. 2016

[15] N. Sambo *et al.*, "Monitoring Plane Architecture and OAM Handler," *J. Lightwave Technology*, vol. 34, no. 8, April 2016, pp. 1939–45.

BIOGRAPHIES

MATTEO DALLAGLIO received his Master's degree cum laude in telecommunication engineering from the University of Trento together with a Diploma in communication network engineering from Scuola Superiore Sant'Anna, Pisa, in 2013. He is currently pursuing a Ph.D. in photonic technologies from Scuola Superiore Sant'Anna. His research interests include software defined networking, network functions virtualization, WDM network planning and modeling, PCE, and GMPLS protocols for traffic engineering.

NICOLA SAMBO is an assistant professor at Scuola Superiore Sant'Anna. He was a visiting student at France Télécom, Lannion. He is also collaborating with CNIT, Pisa, Italy. His research interests include optical network architecture, transmission performance modeling, and control plane. He is an author of about 100 publications including international journals, conference proceedings, and patents.

FILIPPO CUGINI is with CNIT. His main research interests include theoretical and experimental studies in the field of optical communications and networking. He is a co-author of 12 patents and more than 200 international publications.

PIERO CASTOLDI [SM] has been an associate professor at Scuola Superiore Sant'Anna since 2001, where he leads the area of Networks and Services. He had leading roles in the following EU projects: BONE, STRONGEST, IDEALIST, OFELIA, FED4FIRE, and 5GEx. His research interests cover network architectures, control, and data center architectures for grids and clouds. He is an author of more than 300 publications in international journals, conference proceedings, and patents.