Pre-programming resilience schemes upon failure through NETCONF and YANG

M. Dallaglio[1], **N. Sambo**[1], **F. Cugini**[2], **P. Castoldi**[1]
1: Scuola Superiore Sant'Anna, Pisa, Italy; 2: CNIT, Pisa, Italy
matteo.dallaglio@sssup.it

Abstract: We propose and successfully implement a method to program resilience schemes (e.g., code adaptation) in a transponder controller. YANG models are proposed and demonstrated to configure actions and finite state machine in the transponder controller.
OCIS codes: 060.4250, 060.4261.

1. Introduction

Networks are evolving toward higher programmability, making the setup of network devices more flexible and configurable according to the current needs (e.g., proper transmission parameters) [1, 2]. Control and management have to support such programmability. NETCONF is emerging as a Software Defined Network (SDN) protocol for the control and management of optical networks [3]. NETCONF may exploit YANG data modeling to describe in a formalized and standardized way the network device's [4]. In particular, YANG and NETCONF are suitable to describe data plane capabilities, such as the properties of a transponder (e.g., supported bit rate values) [5], or, in general, the structure of an optical network in a *vendor independent way* [6, 7]. This is considered to be very relevant given the increased interest of operators in *white boxes* (i.e., nodes composed of an aggregation of components from different vendors). YANG is organized in a hierarchical structure with `leaves` and `sub-leaves`, each one used to define `data` with a specific type (e.g., integer). As an example, the leaf `bit rate` of a transponder YANG model can be used by the transponder controller to inform the centralized SDN controller about the supported bit rate values [5] or can be used by the SDN controller to set the transponder at the selected bit rate. Currently, YANG does not support the way to instruct a transponder (or in general a data plane device) to self-readapt transmission parameters upon events as failures. Thus, once a failure or a soft failure (i.e., degradation of the bit error rate —BER — due to a network device malfunction) is detected, the device controller has to inform the SDN controller, which performs computations (e.g., selection of new transmission parameters such as modulation format and code) and sets the devices accordingly. Such procedure requires certain time due to the two-ways message exchange between the device and SDN controller and may imply an high load at the centralized controller when many connections are involved. In conclusion, currently, YANG does not define "events" (e.g., soft failure), "actions" (e.g., code adaptation), or "finite state machines", that can be required to enable a centralized SDN controller to instruct a network device to self reconfiguring according to specific events (e.g., failures), and more in general to (re)program the state machine on the device.

In this paper we propose a method — based on novel YANG Models, namely `Events` and `Finite State Machine` — that allows to extend and increase the level of programmability of the network devices. This method allows the centralized controller to instruct the device about the actions to perform once a specific event occurs. More in general, we provide the possibility to configure on the device a finite state machine (FSM) through NETCONF and YANG. The proposed method is finally validated through an experiment in a control/management plane testbed.

2. Pre-programming resilience schemes in a network device

YANG models for events, actions, and state machine are here proposed to enable a SDN controller (on behalf of a network operator) to instruct a device controller about critical events and actions to be taken if these events occur (e.g., increase of redundancy once BER passes a given threshold). The actions to be taken and the critical events can be re-programmed on the device by simply sending a new message configuration (e.g., NETCONF protocol) 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 speed up derives from the fact that the SDN controller is able to pre-configure on the network devices the precise actions to perform when an event occurs. In this way, the device (provided by any vendor) already knows what to do and it can immediately react, avoiding to inform the SDN 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 is notified about the faults and the taken action.

978-1-943580-23-1/17/$31.00 ©2017 Optical Society of America

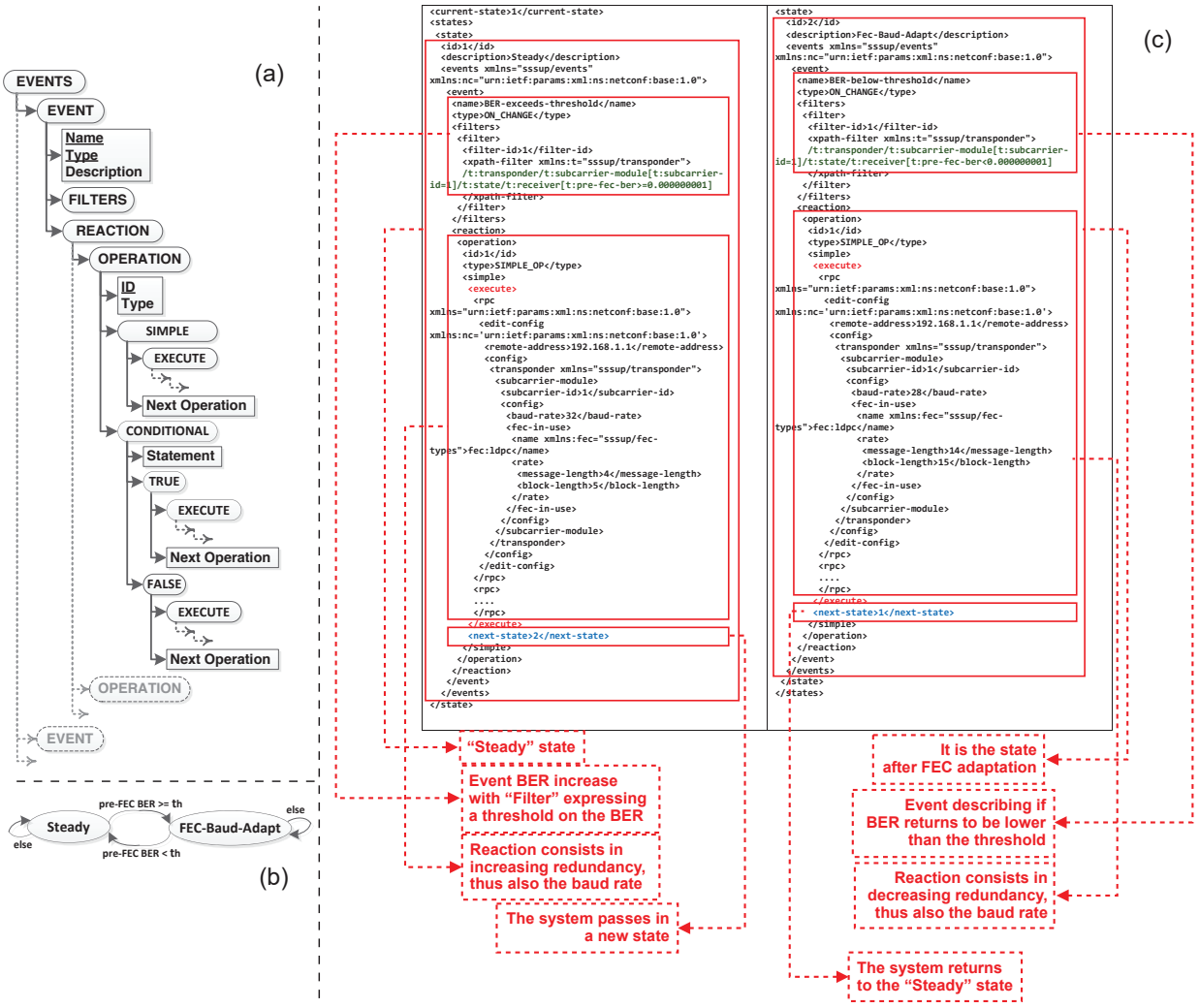

Fig. 1. Events YANG model's tree (a); FSM configured in the experiment (b); Message capture in the experiment (c).

The proposed Events YANG model is schematized by the tree diagram of Fig. 1(a). The model defines a list of events as a root element 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 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, which is fired when the value of an attribute changes. Depending on the type of event some additional attributes (e.g., filters) may be present in order to further express an event. For example we may define an event named *Q-factor change* of type `ON CHANGE` and as filter a threshold to indicate when the Q-factor falls below the threshold. Another fundamental feature proposed in the model is provided by the `Reaction` attribute. In particular, for each event the SDN controller can configure a reaction the device should take. A `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 type `Simple` or `Conditional`. A `Simple` operation contains the `Execute` attribute which 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 and the containers `True` and `False`. The `Statement` is checked at the begin of the operation then, depending on the outcome (true or false), only the correct container is considered. `True` and `False` contains the `Execute` and `Next operation` as for the `Simple` operation.

We also propose a YANG model for a finite state machine (FSM) where each state is based on the Event YANG

model. In particular, the FSM YANG model extends the Events YANG model by adding the state information and state transition. More precisely, the model defines a list of states that, similarly to the events, are configurable by the controller. Each state has a `Description` attribute identified through an `ID`. Each state also includes a list of events as defined in the Events model, with the additional `Next-state` attribute pointing the next state.

3. Experimental demonstration

The proposed method based on the novel YANG models has been experimentally evaluated in a testbed composed of three PCs, one acting as SDN controller and the others two emulating a transponder device at the transmitter and receiver side. The SDN controller runs a python implementation of a NETCONF client. The devices run ConfD, a NETCONF server implementation made by Tail-f (CISCO), and a C program we made to emulate the transponder capabilities. The experiment consists on creating a new connection, configuring the FSM depicted in Fig. 1(b) on the receiver, and emulating a BER change. In particular, we considered a 100-Gbps net rate connection with a baud rate of 28 Gbaud, 7% of forward error correction (FEC) scheme, and polarization multiplexing quadrature phase shift keying (PM-QPSK) as modulation format. The connection is configured by the SDN controller issuing a NETCONF `<edit-conf>` message as detailed in [5]. The FSM 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 threshold $th = 9 \times 10^{-4}$. In case the pre-FEC BER exceeds th, the FSM evolves to the `FEC-Baud-Adapt` state where an adaptation to a more robust FEC (20%) and a baud rate change (to 31 Gbaud) are performed. From the `FEC-Baud-Adapt` state, in case the pre-FEC BER returns below the threshold th, FSM moves back to the `Steady` state readjusting the baud rate and the FEC to the values initially configured. Fig. 1(c) shows a portion of the message sent by the SDN 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 that triggers a change of the baudrate 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. Similarly, to the `Steady` state, a single operation is executed as reaction. In this case, the FEC and the baudrate are restored to the initial values. After FSM configuration, pre-FEC BER above threshold was emulated thus triggering the reaction shown in Fig. 1(c) and implying the transition to the state `Fec-Baud-Adapt`. Then, pre-FEC BER change below threshold was emulated triggering transmission parameter adaptation and transition to the `Steady` state. It is 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).

4. Conclusions

This paper proposed a method to program resilience schemes (e.g., FEC adaptation) in a device controller, e.g. of the transponder. The method is based on proposed YANG models allowing the SDN controller to configure a finite state machine in the transponder controller. The method is validated in a control plane testbed with NETCONF protocol.

5. Acknowledgements

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

References

1. N. Sambo and et al., “Next generation sliceable bandwidth variable transponders,” *Communications Magazine, IEEE*, vol. 53, no. 2, pp. 163–171, Feb 2015.
2. E. Hugues-Salas and et al., “Next generation optical nodes: The vision of the european research project idealist,” *IEEE Communications Magazine*, vol. 53, no. 2, pp. 172–181, Feb 2015.
3. R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, “Network configuration protocol (NETCONF),” IETF RFC 6241, June 2011.
4. M. Bjorklund, “YANG - a data modeling language for the network configuration protocol (NETCONF),” IETF RFC 6020.
5. M. Dallaglio, N. Sambo, J. Akhtar, F. Cugini, and P. Castoldi, “YANG model and NETCONF protocol for control and management of elastic optical networks,” in *Optical Fiber Communication Conference*, 2016.
6. J. Vergara and et al., IETF draft-vergara-flexigrid-yang-00, Oct. 2014.
7. A. Shaikh, T. Hofmeister, V. Dangui, and V. Vusirikala, “Vendor-neutral network representations for transport sdn,” in *2016 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2016, pp. 1–3.