Autonomic Disaggregated Multilayer Networking

Lluís Gifre, Jose-Luis Izquierdo-Zaragoza, Marc Ruiz, and Luis Velasco

***Abstract*—Focused on reducing capital expenditures by opening the data plane to multiple vendors without impacting performance, node disaggregation is attracting the interest of network operators. Although the SDN paradigm is key for the control of such networks, the increased complexity of multilayer networks strictly requires monitoring/telemetry and data analytics capabilities to assist in creating and operating *self-managed* (*autonomic*) networks. Such autonomicity greatly reduces operational expenditures, while improving network performance. In this context, a monitoring and data analytics (MDA) architecture consisting of a centralized data storage with data analytics capabilities, together with a generic node agent for monitoring/telemetry supporting disaggregation is presented. A YANG data model that allows to clearly separate responsibilities for monitoring configuration from node configuration is also proposed. The MDA architecture and YANG data models are experimentally demonstrated through three different use cases: *i*) *virtual link creation* supported by an optical connection, where monitoring is automatically activated; *ii*) *multilayer self-configuration* after BER degradation detection, where a modulation format adaptation is recommended to the SDN controller to minimize errors; this entails reducing the capacity of both the virtual link and the supported MPLS-TP paths; and *iii*) *optical layer self-healing*, including failure localization at the optical layer to find the cause of BER degradation. A combination of active and passive monitoring procedures allows to localize the cause of the failure, leading to lightpath rerouting recommendations toward the SDN controller avoiding the failing element(s).**

***Index Terms*—Autonomic networking, active and passive monitoring, disaggregated multilayer networks.**

I. Introduction And Motivation

Node disaggregation (*whitebox*) focuses on radically reducing capital expenditures (CAPEX), allowing to create real multi-vendor networks, which at are not just based on control plane interoperability [3], without compromising features and/or network performance. Different levels of disaggregation might be conceived, either at the hardware or software level or both simultaneously; for the sake of generalization, we denote as *node* a self-contained element exposing certain level of programmability to Software-Defined Networking (SDN) controllers.

Today, packet-layer whiteboxes are becoming a reality, where extensions for optical devices are still in progress [4]. Several initiatives are working on the definition of optical interoperability in Wavelength Division Multiplexing (WDM) and Elastic Optical Networks (EON) scenarios [5], to produce specifications and YANG data models [6], such as OpenROADM [7] and OpenConfig [8]. However, it is unclear how to operate a disaggregated network and whether such disaggregation will increase the complexity of the network thus, increasing operational expenditures (OPEX). Here, the role of monitoring and telemetry is gaining traction as an enabler for real-time self-managed (*autonomic*) networking [9]. Such autonomic networks will reduce human intervention ultimately reducing OPEX; since network performance information based on measurements is used to identify, isolate, and solve potential issues as quickly as possible, networks will run smoothly. Then, to build autonomic networks, monitoring and telemetry data must be not only gathered from networking devices, but also analyzed for Knowledge Discovery from Data (KDD), so that *recommendations* to the SDN controller can be produced aiming at improving network performance.

In this regard, passive monitoring techniques are the most-common choice in optical networks, since they entail using non-invasive methods to obtain measurements; examples include measuring Bit Error Rate (BER) in optical transponders, optical power in fiber links, as well as acquiring the optical spectrum in the whole C-band in links using optical spectrum analyzers (OSAs). Nonetheless, some active monitoring techniques are also available at the optical layer; for instance, the authors in [10] proposed to use in-band Optical Supervisory Channel (OSC) devices to monitor BER of 100 Gb/s Dual Polarization Quadrature Phase Shift Keying (DP-QPSK) lightpaths at transit points, i.e., not in the transponders. This technique consists in over-modulating the target lightpath with low-speed low-bandwidth On-Off Keying (OOK) signal in an OSC_{TX} device at the ingress node. Then, the BER of the OOK signal is measured at intermediate locations by OSC_{RX} devices using low-speed components, and the BER of the lightpath is estimated through BER correlation curves obtained *a priori*. A similar technique was recently used in [11] for lightpath commissioning testing.

Regarding architectures supporting monitoring [12], authors in [13] define a distributed data analytics framework so-called CASTOR including *extended nodes*, which run close to the network nodes, and a big data centralized *Monitoring and Data Analytics* (MDA) system running in the control and management plane. Extended nodes collect monitoring data records from configured *observation points* (OP) in the nodes, which can be aggregated therein to reduce the amount of data

Manuscript received December 30th, 2017. This work was presented in part at OFC [1] and [2].

Lluís Gifre is with Universidad Autónoma de Madrid (UAM), Madrid, Spain. Jose-Luis Izquierdo-Zaragoza, Marc Ruiz, and Luis Velasco (lvelasco@ac.upc.edu) are with the Optical Communications Group (GCO), at Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.

sent toward the MDA system, and for KDD to proactively notify the MDA system about network anomalies and degradations. The result from local KDD processes could be additionally exploited for network-wide control loops, including recovery and re-optimization [14],[15].

In this paper, we extend the architecture in [13]. The MDA system includes the operational databases (e.g., topology and connections) retrieved from the SDN controller, which entails having a complete view of the network. A generic multi-node agent (*hypernode*) has been conceived to support multilayer disaggregated scenarios; it manages monitoring configuration, data collection, as well as KDD. An autodiscovery function allows retrieving the available monitoring capabilities for each node, as well as already configured OPs. This novel functionality is of paramount importance in brown-field scenarios, where the network is already in operation. Particularly, the contribution of this paper is three-fold:

1) The reference multilayer MPLS-TP-over-optical network architecture is first introduced in Section II, where intra-layer and inter-layer reference interfaces are defined. Three types of nodes are considered to create a partial disaggregated data plane: MPLS-TP switches, optical transponder nodes, and optical switches. The reference interfaces allow defining not only a multilayer network topology, but also physical intra-layer and logical inter-layer connectivity. To this respect, reference interfaces provide primitives to monitor both themselves and associated connections, which can be also enriched with measurements from dedicated monitoring devices like OSAs and OSCs.
2) Section III is devoted to monitoring and data analytics, where the proposed architecture is detailed. A hierarchy of monitoring modules allows bringing distributed and centralized data analytics to the network. A YANG data model specifically designed to manage monitoring, allowing to separate responsibilities when accessing to the node controllers, is proposed. Finally, generic workflows putting together all elements at data/control/management planes are presented.
3) To demonstrate the concepts proposed in the paper three use cases are defined in Section IV, built on top of the workflows presented in Section III. First, a virtual link supported by an optical connection is created. Next, a certain degree of BER degradation in the optical connection is detected and two different workflows are presented for: *i*) multilayer self-configuration; and *ii*) optical layer self-healing, including a failure localization procedure based on the combined use of passive and active optical monitoring/telemetry.

The discussion is supported by the experimental demonstration presented in Section V.

II. Multilayer Networks: Interfaces and Monitoring

As aforementioned, we consider a MPLS-TP-over-optical multilayer network, where the partially disaggregated data plane consists of three elements nodes: the MPLS-TP (L2) switch, the Transponder (TP) node, and the optical (L0) switch. For the sake of simplicity, we denote as *L2* (electronic) both, the Ethernet and the MPLS-TP network sub-layers, whereas we denote as *L0* the whole optical layer, which includes the channel sub-layer (TPs) as well as multiplex/transmission sub-layers (L0 switches) [16].

Let us now define the interfaces within the multilayer network architecture. Three types of interfaces are generically considered in every network (sub-) layer (Fig. 1a): *i*) *server interfaces* (SI), providing access to a given network layer; *ii*) *network interfaces* (NI), connecting nodes in the same network layer. NIs can be physical or logical interfaces as will be described below; and *iii*) *client interfaces* (CI), that access the server network layer. Pairs CI-SI between network (sub-) layers, as well as NI-NI in the same network (sub-) layer are connected through an inter-layer or network layer link, which creates the multilayer network topology.

Regarding inter-layer adaptation, we assume that it is performed inside the network nodes; specifically, Ethernet to MPLS-TP and MPLS-TP to the optical layer adaptation are performed inside the MPLS-TP switches, whereas optical (gray) to WDM (colored) adaptation is performed in the optical transponders inside TP nodes. Finally, network connections (LSPs) are created between two nodes in the same network (sub-)layer (Fig. 1b).

Fig. 1c shows an example of a multilayer partially disaggregated data plane. The L2 switch includes Ethernet interfaces (L2-SI), being the access of client traffic to the network. Ethernet frames are tagged creating L2-LSPs that are switched and leave the switch through outgoing L2 virtual links (*vlink*); note that in such case end NIs are logical interfaces. To access the network, other Ethernet switches or

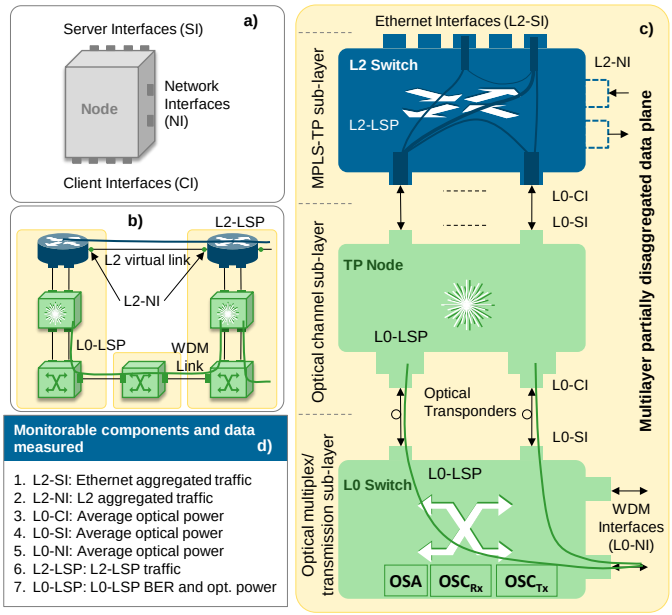

Fig. 1. Multilayer partially disaggregated data plane architecture, interfaces, and monitorable data.

even IP routers might exist being connected to the MPLS-TP switch through Ethernet client interfaces (L2-CI). The TP node integrates optical transponders, so vlinks are supported by optical connections, i.e., L0-LSPs, in the optical layer.

In the presented model, outgoing L2 traffic leaving the L2 node through a given vlink is first aggregated and then adapted to the optical layer and sent through a L0-CI toward the TP node. The optical signal is converted back to a flow in the electrical domain and injected into an optical transponder that converts the flow to a WDM signal using a pre-determined modulation format. The WDM signal is sent to the collocated L0 switch, which includes Wavelength Selective Switches (WSS) and forwards the optical signal through the corresponding WDM interface toward a neighboring L0 switch. When the WDM signal arrives to the remote location, the inverse procedure is performed to bring the electronic flow to the egress L2 switch. From the perspective of the L2 layer, the two end L2 switches appear connected through a L2 link (the vlink).

Each of these interfaces and network layer connections are denoted *monitorable components* as they can provide monitoring data regarding their performance, when observation points (OP) are activated on them. Table in Fig. 1d details the data that can be retrieved from any of the monitorable components. Specifically: *i*) incoming/outgoing Ethernet traffic can be monitored in L2-SIs; *ii*) L2 traffic for the corresponding LSP can be monitored in L2-LSPs; *iii*) aggregated L2 traffic can be monitored in L2 vlink endpoints (L2-NIs); *iv*) average optical power can be monitored in L0 interfaces; and *v*) bit error rate (BER), power, and other parameters of the corresponding L0-LSP can be monitored at the subcarrier module of the transponders.

In addition to transmission and switching devices, network operators are deploying specific monitoring devices that enrich the available monitoring data, as mentioned above. Examples include OSAs and OSCs at the optical layer. Although other configurations are possible without loss of generality, in this paper and we assume that such optical monitoring devices are deployed inside L0 switches to maximize its utilization. In such case, optical spectrum can be obtained from L0-NIs, L0-SIs, and L0-LSPs inside L0 switches, whereas in-line BER can be measured in L0-LSPs also inside L0 switches.

In the case of OSAs, the spectrum of the optical links can be acquired and used to analyze the signal of every single L0-LSP, as proposed in [11]. Algorithms analyzing the optical spectrum can find *i*) deviations in the central frequency, which might be as a result of a laser drift problem in the TP node; *ii*) distortions in the shape of the measured signal, which might be as a consequence of a filter problem in an intermediate L0 switch. Therefore, a simple failure localization procedure can be implemented by retrieving signal diagnosis information from those node-wide algorithms analyzing the spectrum in every L0 switch along the route of an L0-LSP.

A different failure localization procedure can be implemented using OSC devices. By over-modulating the incoming optical signal in the first L0 switch with an OSC_{TX} and measuring the BER in the OSC_{RX} in every other intermediate L0 switch, an algorithm can follow the evolution of the measured BER and compare it against the expected value obtained using an analytical model (e.g., using [17]). Discrepancies between the measured and the expected BER might help localizing failures. The difference between using OSAs and using OSC lies in the way resources are used for the measurements; OSAs are periodically acquiring the optical spectrum of links, whereas a set of OSCs in L0 switches along the route of a given L0-LSP need to be allocated to perform BER measurements. In fact, OSC-based measurement creates continuous streams of BER data that is collected using telemetry.

In conclusion, an MDA system is required to manage networking and monitoring devices, collect measurements and analyze data.

III. Monitoring and Data Analytics

In this section, we present our architecture to support active and passive monitoring, telemetry, and data analytics assuming partially disaggregated multilayer scenarios. A new YANG data model will be needed to facilitate autodiscovery, monitoring and telemetry operations, including active and passive schemes. Operation of the proposed architecture and data model will be eventually described.

A. Architecture

The proposed monitoring and data analytics architecture shown in Fig. 2 consists of a multi-node agent, referred to as *Hypernode*, managed by the centralized MDA system in the control and management plane. The number of hypernodes may vary depending on the size of the network, geographical extension, and/or any other criteria.

Hypernodes are designed to configure monitoring and telemetry, as well as to collect measurements from one or more nodes in the disaggregated data plane. While each node controller usually controls one single node and exposes one single interface toward the SDN controller, hypernodes are designed to be in charge of monitoring and telemetry of a set of nodes. Hypernodes augment *extended node*s from [13] and consist of two building blocks, the *local configuration* module and the *local KDD* module. The local configuration module is in charge of receiving configuration and exposes a RESTCONF-based [18] North-Bound Interface (NBI) to the MDA system. The NBI is based on a YANG data model that includes two subtrees: *i*) *applications*, for configuring the applications in the local KDD module inside the hypernode and *ii*) *nodes*, for configuring the underlying nodes. Finally, a number of node adapters (one per node) are used to implement the specific protocols exposed by every node controller for node configuration and for monitoring data collection.

Although we assume that node controllers' NBIs are based

on a common YANG data model, different protocols for configuration, monitoring, and telemetry might be considered (e.g., NETCONF [19], IPFIX [20], and gRPC-based protocols [21] for configuration, monitoring and telemetry, respectively, NETCONF for everything, or any other combination). For this very reason, node agents include bespoke node adapters implementing specific protocols and function mapping for the underlying node controller.

Regarding the local KDD module, its original scope in the *extended nodes* was to apply data analytics to the measurements received from the nodes focused on the KDD process. Output of the data analytics procedure was forwarded to the MDA system to implement network-wide control loops; two types of messages are supported: *i*) IPFIX-based monitoring messages including processed monitoring samples (i.e., values are averaged over the selected monitoring period, e.g., 15 minutes); and *ii*) through asynchronous notifications using the RESTCONF NBI. Regarding telemetry, measurements are locally processed by specific KDD processes in the hypernode to reduce data exchange with the MDA system. Note that telemetry measurements might be taken in a sub-second basis, so analysis is performed locally in the hypernode and results can be conveyed to the MDA system for decision making.

Similarly to the hypernode, the MDA system has been extended from a similar module described in [13]. Extensions are related to the configuration of the monitoring and telemetry, defined in the new YANG data models. In addition, operational databases (e.g., topology and LSP databases) synchronization from the SDN controller is supported. Full database synchronization, as well as topology or LSP update notifications are now supported. With operational databases synchronized, the CASTOR system is able not only to collate measurements from the nodes (as in [13]), but also to correlate them with the route of a LSP for failure localization purposes. Therefore, more sophisticated procedures can be developed correlating measurements with topology and LSP data.

B. YANG data model and monitoring data

Fig. 3 presents the structure of the proposed YANG data model that is assumed to be supported by every node controller. The model defines two differentiated subtrees with the objective to separate configuration from monitoring and telemetry responsibilities; the SDN controller is focused on the *configuration* subtree, whereas hypernodes are mainly focused on the *monitoring* subtree. The configuration subtree includes every programmable or monitorable component in the node, whilst the monitoring subtree includes monitoring capabilities and OPs and it is specifically designed to facilitate autodiscovery by hypernodes.

A key element in the model is the *component*, representing any configurable or monitorable element in the node and is locally identified by its *component-id*. Nodes are assumed to feed a data store compliant with this model during bootstrapping, whereas dissemination toward hypernodes and SDN controller can be made per-update notifications or by polling. Although components under the configuration and monitoring subtrees are related, we relax such condition to allow obtaining only the monitoring subtree during the autodiscovery process. In fact, the configuration subtree is not stored in the hypernode to avoid synchronization issues; whenever configuration of some component is required by a KDD application, the local configuration module retrieves it directly from the node controller.

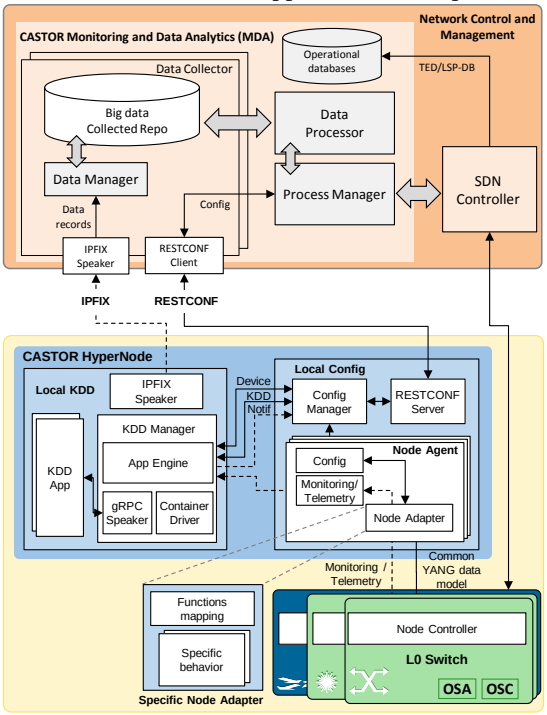

Fig. 2. CASTOR architecture and interfaces.

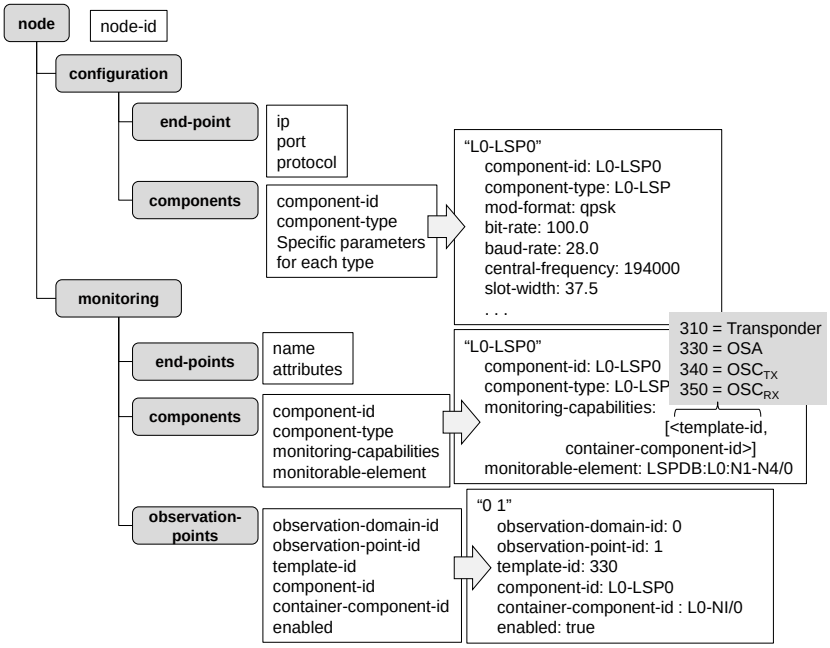

Fig. 3. Proposed YANG data model and examples.

Monitoring/telemetry in monitorable components can be activated by creating and enabling OPs and deactivated by disabling and/or deleting OPs. Monitorable components include a list of tuples with supported monitoring template identifiers (*template-id*) and the container, e.g., an interface, where measurements will be done (*container-component-id*). Monitoring templates represent different measurement methods, so each OP is associated to one single monitoring template and interface tuple. We use the *template-id* to identify the monitoring method used for the measurements, as well as their data structure; specifically, we define the following identifiers: *i*) *template-id* 310 identifies monitoring of BER and optical power in the transponders; *ii*) *template-id* 330 identifies optical spectrum monitoring performed by OSAs, which send sequences of tuples <frequency, optical power>; *iii*) *template-id* 340 identifies OSC_{TX} devices that report heartbeat/keepalive messages; and *iv*) *template-id* 350 identifies OSC_{RX} devices that send estimated BER values.

The monitoring subtree includes also every existing OP (enabled, e.g., performing measurements or disabled). Specifically, the ability to enable and disable existing OPs supports allocation of monitoring devices in active monitoring schemes. Nodes are expected to allocate monitoring resources by translating tuples <*component-id, template-id, container-component-id*> into device-specific commands. Observation domain/point identifiers are internal identifiers used by KDD processes inside hypernodes to isolate different datasets, and they are included in monitoring messages. Multiple OPs can be defined for a single component, each reporting monitored data formatted as per the selected supported template.

Fig. 3 presents an example for a lightpath (L0-LSP): configuration parameters include, but are not limited to, modulation format, bit/baud-rate, and the allocated frequency slot specified by *central-frequency* and *slot-width* parameters. In the monitoring subtree, components include two relevant attributes: *i*) *monitorable-element* ("what"), which is used for correlation purposes between the operational databases in the SDN controller and the MDA system; and *ii*) *monitorable-capabilities* containing a set of pairs with the supported *template-id* ("how"), representing the monitoring method that such component supports in the specific container component, identified by its *container-component-id* in the network node ("where"). For illustrative purposes all the templates have been included in Fig. 3, but only those supported by the component in the network node will be advertised. Finally, an OP is currently active; specifically, L0-LSP0 is being monitored in interface L0-NI/0 using an OSA.

C. Operation

Fig. 4 presents the three basic workflows supported: *i*) *autodiscovery*, *ii*) *LSP set-up and monitoring / telemetry activation*, and *iii*) *KDD process*. These basic workflows will be part of the use cases presented and experimentally demonstrated in the following sections. Additionally, the workflows give insight of the internal relations in the hypernodes, which are represented by the local KDD and the local config internal blocks. Each exchanged message is identified with a number that is relative to the workflow it belongs to; note that numbers are reset in every workflow.

The autodiscovery workflow is initiated by the CASTOR MDA system at startup, when the operational databases are retrieved from the SDN controller, as well as by the SDN controller after a new node has been added to its topology database. In the second case, a notification is sent to the CASTOR MDA system (message 1), which updates its topology database and request the corresponding hypernode to start the autodiscovery process (2); specifically, the *node-id* and the parameters for configuration, monitoring, and telemetry are needed to start the autodiscovery process. Upon the reception of the start autodiscovery message, the local configuration block in the hypernode instantiates a new node agent with the specific adapter, which connects to its peer node controller to get the current *Monitoring* subtree status (3) that is sent back to the MDA system (4).

The LSP set-up and monitoring/telemetry activation workflow is triggered by a request to the SDN controller (message 0). After computing the specific routing algorithm (e.g., the routing, modulation, and spectrum allocation - RMSA-algorithm for L0-LSPs [14]), the SDN controller sends configuration messages (1) to the node controllers along the route of the LSP specifying a set of parameters, e.g., the spectrum allocation in the case of a L0-LSP; additionally, the LSP symbolic name is added to the configuration message so it can be used as identifier in future messages. When the LSP has been finally set-up, a notification is sent to the CASTOR MDA system (3) containing the route of the LSP, configuration parameters and symbolic name; however, the LSP needs to be also discovered by the hypernodes so as to ensure the proper data synchronization. To that end, node controllers notify the corresponding hypernodes about resources allocated for a new LSP, identified by its local identifier and the identifier received from the SDN controller (2). In this case, the hypernode needs to obtain from the node controller the monitoring subtree for the specific LSP and the node controller replies with the set of pairs <*template-id*, *container-component-id*> supported for such LSP (4). Next, the hypernode sends a notification toward the MDA system announcing that a new LSP has been discovered; the notification includes the SDN controller identifier for the LSP, the local identifier and the monitoring capabilities (5); the notification is correlated with the one received from the SDN controller. Next, the MDA system creates and activates a subset of OP) for the LSP (6). Requests to create and activate OPs are sent to the node controllers (7). In the local KDD, OP handlers, called Observation Groups (OG) are created inside a KDD application to collect, analyze, and aggregate measures from a set of OPs (8); once an OG has been created, add/remove OPs messages can be issued. The hypernode replies the MDA system once the creation and activation processes are completed (9).

Once OPs are activated, monitoring/telemetry data is

received and can be analyzed to discover patterns. The KDD process workflow starts when new measurements are received in the hypernode (message 1); monitoring data records are analyzed by the corresponding process in a KDD application and, when a certain pattern is detected, the process asks the

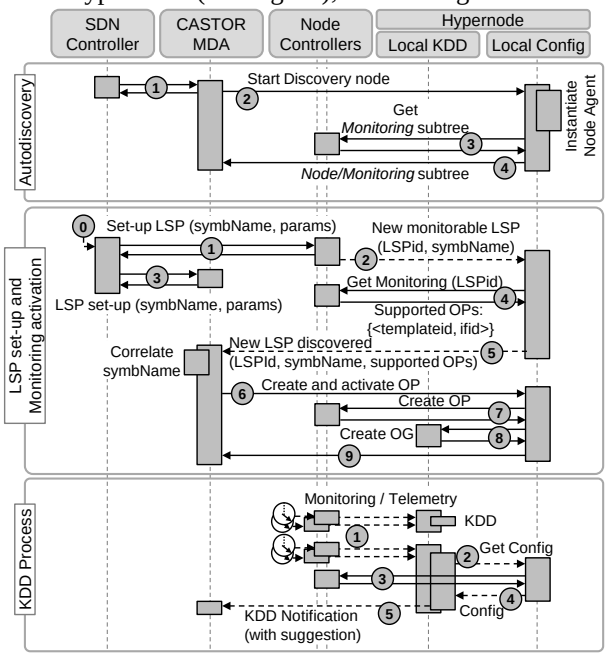

Fig. 4. Basic workflows supported by hypernodes.

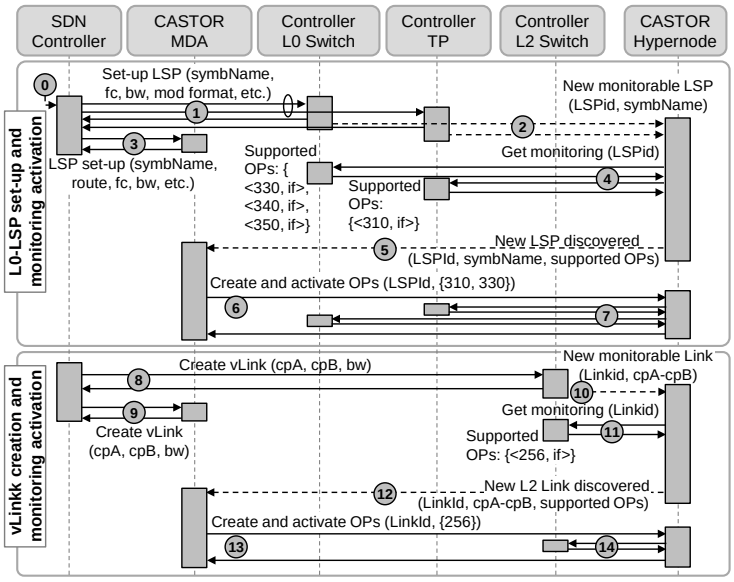

Fig. 5. WF1: L2 virtual link creation supported by a L0-LSP.

local config block in the hypernode (2) to interrogate the node controller about the current configuration of some components (3). The KDD process analyzes the received component configuration to find whether it can be modified and notifies the MDA system about the discovered pattern; a suggestion of parameter tuning might be included in the notification as a result of the configuration analysis (5). With such notification, the MDA system can make decisions, including suggesting a reconfiguration to the SDN controller.

IV. Use Cases

In this section, we consider the architecture proposed above and apply the defined workflows on a multilayer scenario. Here, we assume that the topology has been already discovered and systems are synchronized. The use cases use a L2 vlink supported by a L0-LSP. The evolution of the BER in the supporting L0-LSP is monitored in the hypernode and in case that BER degradation is detected a notification is sent to the MDA system [22]. Two different alternatives are considered once the notification is received in the MDA system: *i*) *L0/L2 self-configuration*, entailing modification of the modulation format for the underlying L0-LSP and, consequently, reducing the available bandwidth in the vlink, as well as in every L2 LSP using such vlink. Note that different policies can be applied in this case, such as rerouting L2-LSPs to keep the allocated bandwidth [23] or even creating a parallel L0-LSP to keep invariant the vlink capacity; and *ii*) *L0 self-healing*, where a failure localization procedure using OSAs and OSCs is executed to localize the optical link responsible for the high BER. Upon completion, the L0-LSP can be restored avoiding that optical link.

A. L2 virtual link creation

The workflow for a L2 vlink creation supported by a L0-LSP (WF1) is presented in Fig. 5. The workflow consists of two phases: first a L0-LSP between two end TP nodes is set-up (messages 0-7) and then, the vlink is created between the two end L2 switches (messages 8-14). Monitoring is automatically activated for both, the L0-LSP and the vlink.

The L0-LSP set-up and monitoring activation phase is based on the generic LSP set-up workflow described in the previous section. The only particularities are that the involved nodes are TPs and L0-switches. The parameters that are included in messages (1) and (3) are those related to the optical layer, such as central frequency (*fc*), slot width (*bw*), modulation format, etc. Regarding monitoring capabilities, the TP node controller reports that the L0-LSP can be monitored in the transponders (*template-id* 310), whereas the L0 switch controller reports that it supports OSA monitoring (*template-id* 330), as well as OSC active monitoring (*template-id* 340 and 350). Finally, the MDA system decides to create and activate OPs related to transponders and OSAs; note that active monitoring entails resource reservation and it will be activated on-demand when needed for failure localization.

The vlink creation and monitoring activation phase is based also on the workflow described in the previous section. In this case, the parameters needed for the creation are the two end points and the capacity (*bw*). In this case, the involved nodes are the end L2 switches that report L2 traffic monitoring capabilities (*template-id* 256) and OPs are automatically created and activated.

B. L0/L2 self-configuration

Once the vlink is in operation, L2 LSPs can be set-up using its available capacity and monitoring data records are collected and analyzed by the hypernodes. Fig. 6 presents the workflow (WF2) for this use case that is divided into two phases: BER degradation detection (messages 1-3 in Fig. 6) and multilayer re-configuration (messages 4-10).

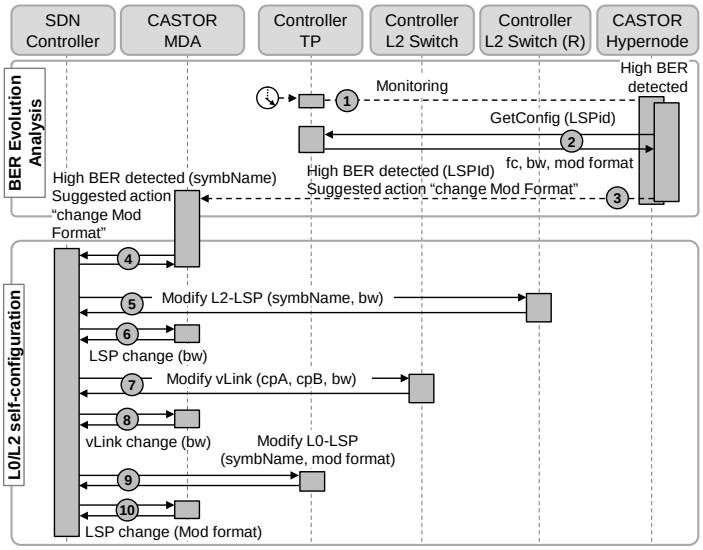

Fig. 6. WF2: High BER detection and L0/L2 self-configuration.

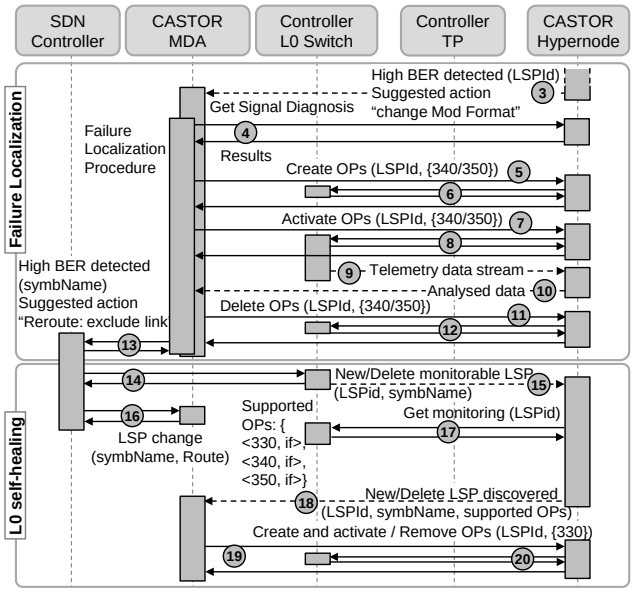

Fig. 7. WF3: Failure localization and L0 self-healing.

The BER evolution is analyzed for the L0-LSP (1), so a workflow similar to the KDD process described in the previous section is followed. In case that BER degradation is detected, the KDD process requests the current configuration to the TP node controller, where in this case includes the optical parameters configured at set-up time, including the modulation format (2). Let us assume that the QAM-16 modulation format is currently used for the L0-LSP; thus, changing the modulation format to QPSK would improve the quality of transmission, which would eventually reduce the BER. In consequence, the KDD process issues a notification to the MDA system reporting the high BER detected and includes the change to the QPSK modulation format as recommended action (3).

In case that upon the notification reception the MDA system follows the recommended action, it sends a notification to the SDN controller recommending the change in the modulation format of the L0-LSP (4). In view of that, the SDN controller follows an internal policy to first reduce the capacity of any L2-LSP using the affected vlink (traffic shaping is re-configured in the ingress L2 switch of the LSPs) (5), then it reduces the capacity of the vlink in the end L2 switches (7), and it finally changes the modulation format of the supporting L0-LSP in the end TP nodes (9). The SDN controller informs about the changes to the MDA system (messages 6, 8, and 10).

C. L0 self-healing

Let us assume now that when the notification with the BER degradation detection (message 3 in Fig. 6) is received in the MDA system, the recommended action to change the modulation format is not followed and a procedure to localize the failure affecting the L0-LSP is triggered instead. Fig. 7 presents the workflow (WF3) for this use case that includes the failure localization procedure at the optical layer (4-13) and the L0-LSP re-routing excluding the identified optical link (14-20).

The failure localization procedure is triggered when the notification with BER degradation detection is received in the MDA system. We assume now that, in contrast to the previous use case, a specific procedure for failure localization is available as one of the data analytics processes in the data processor module in the MDA system (see Fig. 2). The failure localization procedure requests signal diagnosis to every hypernode in charge of a L0 switch in the route of the L0-LSP. A KDD process is in charge of analyzing the optical spectrum of the LSP using the received OSA data records from the local L0 switch (4); the received answers allows the MDA system to determine the cause of the failure (e.g., filter misconfiguration, laser drift, or other), as well as to localize the failure [11]. To confirm the localization of the failure, the procedure might decide to use the OSC monitoring devices to measure the BER along the route of the L0-LSP. To that end, it requests the hypernodes with OSC devices in the route of the L0-LSP to reserve an OSC_{TX} (in the first node) or an OSC_{RX} (in the rest of the nodes) by creating OPs with *template-id* 340/350 in the corresponding container components (5) and in turn, hypernodes request the creation of the OPs (reserving the OSC devices) to the local L0 switch controller (6). When all the confirmations of the devices reservation arrive in the MDA system, OP activation is performed (7 and 8). Note that in this workflow, OP creation and activation is dissociated since monitoring devices need to be first reserved. Once OPs are activated, telemetry is used to send a continuous data stream with BER measurements to the hypernode (9); the corresponding KDD process analyzes the

telemetry data and reports the MDA system (10). The measured BER is compared to the expected BER computed analytically; this allows finding divergences between expected and measured BER, thus localizing the failure. Once the failure is localized, the OPs are deleted, so the OSC devices are released (11 and 12). Assuming that both localization methods return the same failure localization, e.g., in a link due to high noise in an optical amplifier, a notification is sent to

	Source	Destination	Protocol	Info
1	ONOS	CASTOR_MDA	HTTP	POST /rest/newNode HTTP/1.1
2	CASTOR_MDA	ONOS	HTTP	HTTP/1.1 200 OK (application/json)
3	CASTOR_MDA	CASTOR_HN1	HTTP	POST /rest/rpc-autodiscovery HTTP/1.1
	CASTOR_HN1	X1_Ctrl	NetConf	NetConf
	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
4	CASTOR_HN1	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)

Fig. 8. Exchanged messages for X1 L0 switch autodiscovery.

the SDN controller informing about BER degradation in the L0-LSP with the suggested action to reroute the LSP excluding the identified link (13).

Upon the reception of the notification, the SDN controller follows the recommended action and finds an alternative route for the degraded L0-LSP. Note that more than one L0-LSP might be affected by the degradation in the link (this actually depends on the BER thresholds configured for each L0-LSP), so a planning tool might be used to compute optimal rerouting for the set of affected LSPs [23]. Once the route of the LSPs are determined, a similar workflow as the one for LSP set-up is followed and OPs are created in the L0 switches entering in the new route and deleted from those leaving the route.

V. Experimental Results

This section is devoted to the experimental demonstration of the workflows defined in Section IV. The experiments have been carried out in the UPC's SYNERGY testbed. The scenario consists of a CASTOR MDA system and the ONOS SDN controller [24] in the control and management plane and a number of network locations each with an hypernode in charge of one local L2 switch, TP node, and L0 switch. CASTOR MDA and hypernodes, as well as node controller emulators have been developed in Python, where node controller emulators expose a NETCONF NBI based on the YANG data model presented in Section III.B. Communication between MDA system and ONOS is performed through an ad-hoc, bidirectional REST API.

A. Starting procedure and autodiscovery

Before workflows in Section IV can be demonstrated both ONOS and node controllers are started, so that the network is in operation. After starting hypernodes in the network locations and the MDA system in the control and management plane, an autodiscovery workflow (see Section III.C) needs to be carried out for synchronization purposes. This workflow is run each time a new node is added to ONOS. Fig. 8 presents a capture with the exchanged messages for X1 L0 switch autodiscovery; message numbering is consistent with that in Fig. 4. Remote Procedure Calls (RPC) are used in RESTCONF and NETCONF interfaces to facilitate data access, as shown in messages (2-3).

B. Virtual link creation

Fig. 9 shows the exchanged messages for the creation of a L2 vlink between switches in locations 1 and 7; message numbering is consistent with that in Fig. 5, where only messages exchanged for location 1 are shown. Although notifications though the RESTCONF interface (messages 5 and 12 in Fig. 9) are not properly shown in the capture (Wireshark cannot parse HTTP event-streaming), their

	Source	Destination	Protocol	Info
1	ONOS	X1_Ctrl	NetConf	NetConf
	X1_Ctrl	ONOS	NetConf	NetConf
2	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
1	ONOS	TP1_Ctrl	NetConf	NetConf
	TP1_Ctrl	ONOS	NetConf	NetConf
2	TP1_Ctrl	CASTOR_HN1	NetConf	NetConf
3	ONOS	CASTOR_MDA	HTTP	POST /rest/LSP/set-up HTTP/1.1
	CASTOR_MDA	ONOS	HTTP	HTTP/1.1 200 OK (application/json)
4	CASTOR_HN1	TP1_Ctrl	NetConf	NetConf
	TP1_Ctrl	CASTOR_HN1	NetConf	NetConf
	CASTOR_HN1	X1_Ctrl	NetConf	NetConf
	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
5	CASTOR_HN1	CASTOR_MDA	TCP	[TCP segment of a reassembled PDU]
6	CASTOR_MDA	CASTOR_HN1	HTTP	POST /rest/OP/activate HTTP/1.1
7	CASTOR_HN1	TP1_Ctrl	NetConf	NetConf
	TP1_Ctrl	CASTOR_HN1	NetConf	NetConf
	CASTOR_HN1	X1_Ctrl	NetConf	NetConf
	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
6	CASTOR_HN1	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)
8	ONOS	L2Sw1_Ctrl	NetConf	NetConf
	L2Sw1_Ctrl	ONOS	NetConf	NetConf
9	ONOS	CASTOR_MDA	HTTP	POST /rest/vLink/set-up HTTP/1.1
	CASTOR_MDA	ONOS	HTTP	HTTP/1.1 200 OK (application/json)
10	L2Sw1_Ctrl	CASTOR_HN1	NetConf	NetConf
11	CASTOR_HN1	L2Sw1_Ctrl	NetConf	NetConf
	L2Sw1_Ctrl	CASTOR_HN1	NetConf	NetConf
12	CASTOR_HN1	CASTOR_MDA	TCP	[TCP segment of a reassembled PDU]
13	CASTOR_MDA	CASTOR_HN1	HTTP	POST /rest/OP/activate HTTP/1.1
14	CASTOR_HN1	L2Sw1_Ctrl	NetConf	NetConf
	L2Sw1_Ctrl	CASTOR_HN1	NetConf	NetConf
15	CASTOR_HN1	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)

Fig. 9. Exchanged messages for WF1.

```
'exportTime':'2017-Dec-27 18:53:21',
'observationDomainId': 0,
'dataSets': [{
  'setId': 330,
  'records': [{
    'observationPointId': 1,
    'subTemplateList': {
      'semantics': 'ordered',
      'templateId': 10000,
      'records': [{
        'frequencyTHz': 193.98,
        'rxPowerdBm': -60.01492
      }, ..., {
      }] }   }]  }]
```

Fig. 10. Details of OSA monitoring.

contents are properly decoded by the MDA system.

OSAs are continuously acquiring the optical spectrum in the corresponding outgoing link, and when an OP is activated for a L0-LSP (message 7 in Fig. 9), its spectrum is encoded and sent in IPFIX messages using *templateId* 330, which includes a *subTemplateList* field to convey a structured list of data records [20]. This field is really convenient to encode optical spectrum data, since it allows including an ordered list of tuples <*frequency*, *power*>; Fig. 10 presents the details of IPFIX messages for OSA monitoring.

The complete workflow WF1 was executed in less than 1 second, considering message exchange and control and management plane time, i.e., excluding device configuration.

C. L0/L2 self-configuration

Fig. 11 shows the exchanged messages for WF2, when high BER is detected in the L0-LSP supporting the vlink between locations 1 and 7; message numbering is consistent with that in Fig. 6, where messages exchanged for locations 1 and 8 (ingress L2 switch of an L2-LSP using the vlink) are shown.

Fig. 12 plots the evolution of the overall L2 traffic through the vlink, as well as the individual traffic of each (two in this case) L2-LSP traversing it; the capacity of the vlink is also

	Source	Destination	Protocol	Info
1	TP1_Ctrl	CASTOR_HN1	CFLOW	IPFIX flow (52 bytes) [Data:300]
1	TP1_Ctrl	CASTOR_HN1	CFLOW	IPFIX flow (52 bytes) [Data:300]
2	CASTOR_HN1	TP1_Ctrl	NetConf	NetConf
2	TP1_Ctrl	CASTOR_HN1	NetConf	NetConf
3	CASTOR_HN1	CASTOR_MDA	TCP	[TCP segment of a reassembled PDU]
4	CASTOR_MDA	ONOS	HTTP	POST /LSP/performance HTTP/1.1
4	ONOS	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)
5	ONOS	L2Sw8_Ctrl	NetConf	NetConf
5	L2Sw8_Ctrl	ONOS	NetConf	NetConf
6	ONOS	CASTOR_MDA	HTTP	POST /rest/LSP/config HTTP/1.1
6	CASTOR_MDA	ONOS	HTTP	HTTP/1.1 200 OK (application/json)
7	ONOS	L2Sw1_Ctrl	NetConf	NetConf
7	L2Sw1_Ctrl	ONOS	NetConf	NetConf
8	ONOS	CASTOR_MDA	HTTP	POST /rest/vLink/config HTTP/1.1
8	CASTOR_MDA	ONOS	HTTP	HTTP/1.1 200 OK (application/json)
9	ONOS	TP1_Ctrl	NetConf	NetConf
9	TP1_Ctrl	ONOS	NetConf	NetConf
10	ONOS	CASTOR_MDA	HTTP	POST /rest/LSP/config HTTP/1.1
10	CASTOR_MDA	ONOS	HTTP	HTTP/1.1 200 OK (application/json)

Fig. 11. Exchanged messages for WF2.

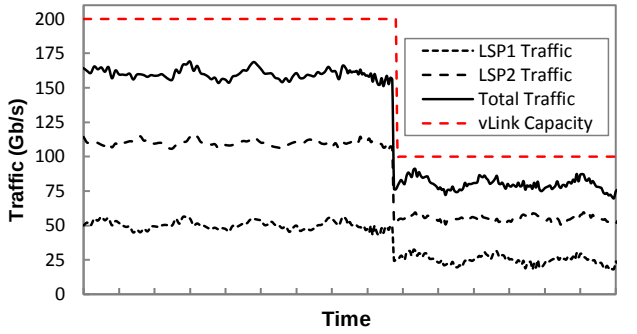

Fig. 12. Virtual link capacity and L2 traffic evolution.

shown for reference. As a result of traffic shaping in the ingress of the L2-LSPs, its traffic is limited to half of the original one, which reduces the total traffic through the vlink. The capacity of the vlink is eventually reduced.

As mentioned in Section IV.B, we are considering an operational scheme in which all L2-LSPs are proportionally reshaped upon L0-LSP capacity changes. It is worth mentioning that bandwidth of any L2-LSPs cannot fall behind the minimum bandwidth guarantees (sometimes referred to as committed information rate, or CIR) according to the associated service-level agreement (SLA) [25]. Otherwise, other mechanisms like L2 rerouting must be used to avoid CIR/SLA violation. In this regard, commercial devices allow different configurations of L2-LSPs depending on how vlink bandwidth is divided among them: absolute (e.g., 10 Gb/s) or proportional (e.g. 10% of vlink capacity). In our use case, we consider that bandwidth of L2-LSPs is defined following the absolute syntax, which is the reason why L2-LSP reconfiguration (5) is performed before vlink reconfiguration (7), to avoid the sum of CIRs to exceed total vlink capacity.

D. L0 self-healing

Fig. 13 shows the exchanged messages for WF3; message numbering is consistent with that in Fig. 7, where only messages exchanged for location 1 are shown. This workflow uses active and passive monitoring techniques for failure localization purposes. Fig. 14a presents the route of the L0-LSP established from TP-1 to TP-7 supporting the vlink between L2Sw1 and L2Sw7. In particular, the architecture of L0 switches is shown in Fig. 14a, where pools of OSAs, each acquiring the whole C-band of an outgoing L0-NI, as well as pools of OSCs with OSC_{TX} and OSC_{RX} are available.

	Source	Destination	Protocol	Info
3	CASTOR_HN1	CASTOR_MDA	TCP	[TCP segment of a reassembled PDU]
4	CASTOR_MDA	CASTOR_HN1	HTTP	POST /rest/LSP/diagnosis HTTP/1.1
4	CASTOR_HN1	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)
5	CASTOR_MDA	CASTOR_HN1	HTTP	POST /rest/OP/create HTTP/1.1
6	CASTOR_HN1	X1_Ctrl	NetConf	NetConf
6	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
5	CASTOR_HN1	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)
7	CASTOR_MDA	CASTOR_HN1	HTTP	POST /rest/OP/activate HTTP/1.1
8	CASTOR_HN1	X1_Ctrl	NetConf	NetConf
8	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
9	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
9	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
7	CASTOR_HN1	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)
10	CASTOR_HN1	CASTOR_MDA	TCP	[TCP segment of a reassembled PDU]
11	CASTOR_MDA	CASTOR_HN1	HTTP	POST /rest/OP/delete HTTP/1.1
12	CASTOR_HN1	X1_Ctrl	NetConf	NetConf
12	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
11	CASTOR_HN1	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)
13	CASTOR_MDA	ONOS	HTTP	POST /rest/LSP/performance HTTP/1.1
13	ONOS	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)
14	ONOS	X1_Ctrl	NetConf	NetConf
14	X1_Ctrl	ONOS	NetConf	NetConf
15	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
16	ONOS	CASTOR_MDA	HTTP	POST /rest/LSP/config HTTP/1.1
16	CASTOR_MDA	ONOS	HTTP	HTTP/1.1 200 OK (application/json)
17	CASTOR_HN1	X1_Ctrl	NetConf	NetConf
17	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
18	CASTOR_HN1	CASTOR_MDA	TCP	[TCP segment of a reassembled PDU]
19	CASTOR_MDA	CASTOR_HN1	HTTP	POST /rest/OP/create HTTP/1.1
20	CASTOR_HN1	X1_Ctrl	NetConf	NetConf
20	X1_Ctrl	CASTOR_HN1	NetConf	NetConf
19	CASTOR_HN1	CASTOR_MDA	HTTP	HTTP/1.1 200 OK (application/json)

Fig. 13. Exchanged messages for WF3.

When the MDA system is notified upon BER degradation detection in TP1 (message 3 in Fig. 13), it decides to run the failure localization procedure before notifying the SDN controller. To that end, the hypernodes along the route of the L0-LSP are interrogated to get a diagnosis of the spectrum of the optical signal received from the OPs activated in the OSAs (4). Fig. 14b-c present two different failure cases, where graphs plotting the optical spectrum were captured in X1, X2, X5, and X3 L0 switches. Fig. 14b reproduces the case where a filter shift failure happens in X5; note that the signal is asymmetrically filtered before the OSA in X5, thus pointing out a misconfiguration of a WSS in such node (WSSs within the red box in Fig. 14a). In this case, a specific procedure could be followed to solve the WSS misconfiguration. Fig. 14c reproduces the case where no filtering problem is detected in the intermediate nodes. In such case, OSAs monitoring cannot localize the failure and OSC-based active monitoring can be used as an alternative.

Recall that OSC-based active monitoring requires to reserve OSC resources in the intermediate L0 switches (i.e., X2, X5, X3, and X7); in particular, one OSC_{TX} is strictly needed. Assuming that OPs are created (OSC resources are reserved) and activated (5-8), a gRPC-based telemetry data stream is created from every L0 switch controller toward the local

hypernode with the estimated BER from the measurements performed on the low-speed signal (9); OSC_{RX} produces BER data records at a sub-second rate. Telemetry data is analyzed by the corresponding KDD process in the hypernode to produce average BER values that are sent to the MDA system (10). Fig. 14d presents an overall chart with BER estimations received from OSC_{RX} and expected BER values computed using analytical models. In view of the different slopes in the segment X5 to X3, it can be concluded that something happens in such optical link (e.g., high noise generated by some optical amplifier).

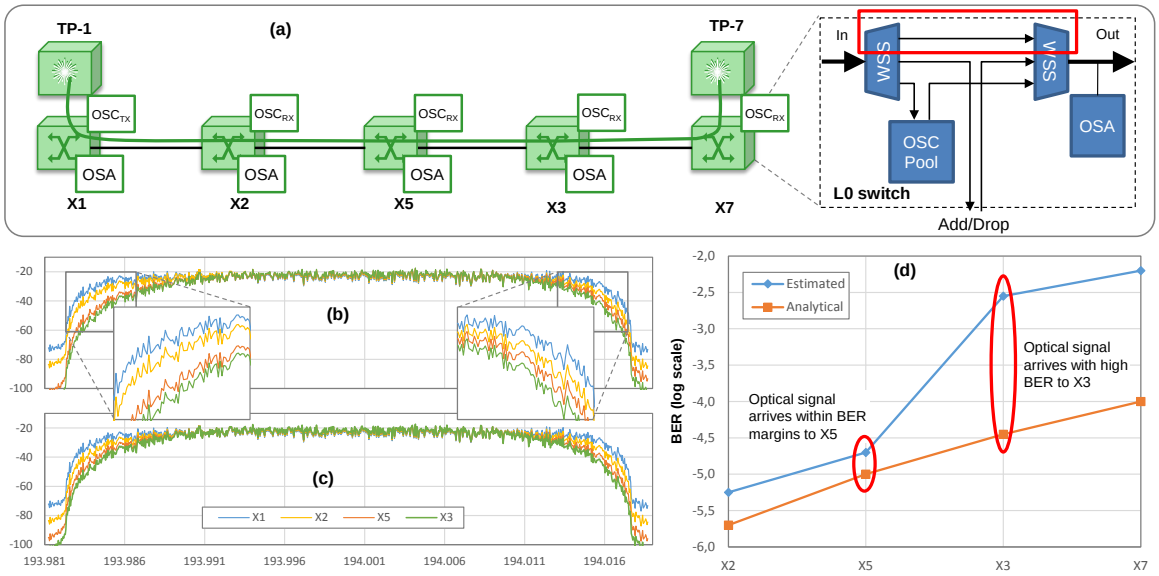

Fig. 14. Illustrative scenario for failure localization: (a) physical route for a reference lightpath; (b) OSA traces for a filter-shift event in X5; (c) OSA traces for an in-line amplifier problem in X5-X3; and (d) OSC measures for the same event as in (c).

VI. Conclusions

An architecture for operation of autonomic operation of multilayer disaggregated networks, has been presented; the architecture is based on the combination of SDN controllers and MDA systems. On the one hand, the SDN controller enables programmability of the network not only in terms of provisioning of connectivity services, but also in terms of reconfiguration. On the other hand, the MDA system provides a distributed platform for network intelligence processes based on monitoring data analytics. Together, these two entities may enable the deployment of production-ready autonomic (self-managed) multilayer networks. Note that hypernodes enable such data analysis distribution, which brings many benefits such as: *i*) the reduction of detection times for failures and anomalies, the reduction; *ii*) the reduction of data conveyed to the centralized MDA system; *iii*) the improvement in the scalability of the control and management architecture, etc.

Here, we consider a disaggregated data plane composed by three types of network devices: MPLS-TP switches, optical transponder nodes, and optical switches. Aiming at identifying candidate components in which measure performance, a reference multilayer MPLS-TP-over-optical network architecture has been introduced, where intra-layer and inter-layer reference interfaces has been defined. Although network devices are usually able to perform measurements on those reference interfaces, dedicated monitoring devices can be used to enrich such measurements.

Besides, the architecture is complemented with a YANG data model specifically designed to manage monitoring, focused on separating responsibilities for monitoring configuration and network configuration when accessing the node controllers. General workflows that combine the above elements were presented, including autodiscovery, LSP set-up and OP creation and activation, monitoring and KDD process.

To demonstrate the concepts presented in the paper, three workflows were defined based on the generic ones and experimentally demonstrated. Firstly, a virtual link supported by an L0-LSP was created. Upon detection of BER in the L0-LSP, two different workflows were presented for multilayer self-configuration and optical self-healing, including a failure localization procedure based on the combined use of passive and active optical monitoring/telemetry using OSA and OSCs. It was demonstrated that OSA and OSC monitoring/telemetry provide complementary views, and their combined use increased accuracy of failure localization.

Acknowledgment

The research leading to these results has received funding from the European Commission for the H2020-ICT-2016-2 METRO-HAUL project (G.A. 761727), from the AEI/FEDER TWINS project (TEC2017-90097-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

References

[1] L. Velasco, Ll. Gifre, and J-L. Izquierdo-Zaragoza, “Toward Multilayer Disaggregated Node Telemetry and Local Decision Making,” in *Proc. of OFC 2018*.

[2] Ll. Gifre, J-L. Izquierdo-Zaragoza, B. Shariati, and L. Velasco, "Experimental Demonstration of Active and Passive Optical Networks Telemetry," in *Proc. of OFC 2018*.

[3] O. González de Dios *et al.*, "Experimental Demonstration of Multi-vendor and Multi-domain Elastic Optical Network with data and control interoperability over a Pan-European Test-bed," *Journal of Lightwave Technology*, vol. 34, pp. 1610-1617, 2016.

[4] R. Nejabati, S. Peng, and D. Simeonidou, "Optical network democratization," *Philosophical Transactions of the Royal Society A*, vol. 374, Aug. 2015.

[5] V. López and L. Velasco, *Elastic Optical Networks: Architectures, Technologies, and Control*, 1st edition, Springer, 2016.

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

[7] OpenROADM: [on-line] www.openroadm.org.

[8] OpenConfig: [on-line] www.openconfig.net.

[9] M. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. Carpenter, S. Jiang, L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals," IETF RFC 7575, 2015.

[10] D. Geisler, R. Proietti, Y. Yin, R. Scott, X. Cai, N. Fontaine, L. Paraschis, O. Gerstel, and S. J. B. Yoo, "Experimental demonstration of flexible bandwidth networking with real-time impairment awareness," *Optics Express*, vol. 19, pp. B736-B745, 2011.

[11] A. P. Vela, B. Shariati, M. Ruiz, F. Cugini, A. Castro, H. Lu, R. Proietti, J. Comellas, P. Castoldi, S. J. B. Yoo, and L. Velasco, "Soft Failure Localization during Commissioning Testing and Lightpath Operation [Invited]," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, pp. A27-A36, 2018.

[12] A. P. Vela, M. Ruiz, L. Velasco, "Distributing Data Analytics for Efficient Multiple Traffic Anomalies Detection," Elsevier Computer Communications, vol. 107, pp. 1-12, 2017.

[13] L. Velasco, Ll. Gifre, J.-L. Izquierdo-Zaragoza, F. Paolucci, A. P. Vela, A. Sgambelluri, M. Ruiz, and F. Cugini, "An Architecture to Support Autonomic Slice Networking [Invited]," *Journal of Lightwave Technology*, vol. 36, pp. 135-141, 2018.

[14] L. Velasco, A. P. Vela, F. Morales, and M. Ruiz, "Designing, Operating and Re-Optimizing Elastic Optical Networks [Invited Tutorial]," *Journal of Lightwave Technology*, vol. 35, pp. 513-526, 2017.

[15] L. Velasco and M. Ruiz, *Provisioning, Recovery and In-operation Planning in Elastic Optical Networks*, 1st edition, Wiley, 2017.

[16] *Architecture of optical transport networks*, ITU-T Recommendation G.872, 2017.

[17] P. Poggiolini, "The GN Model of Non-Linear Propagation in Uncompensated Coherent Optical Systems," *Journal of Lightwave Technology*, vol. 30, pp. 3857-3879, 2012.

[18] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," IETF RFC 8040, 2017.

[19] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," IETF RFC 6241, 2011.

[20] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol," IETF RFC 7011, 2013.

[21] gRPC: A high performance, open-source universal RPC framework: [on-line] https://grpc.io.

[22] A. P. Vela, M. Ruiz, F. Fresi, N. Sambo, F. Cugini, G. Meloni, L. Potì, L. Velasco, and P. Castoldi, "BER Degradation Detection and Failure Identification in Elastic Optical Networks," *Journal of Lightwave Technology*, vol. 35, pp. 4595-4604, 2017.

[23] A. Castro, R. Martínez, R. Casellas, L. Velasco, R. Muñoz, R. Vilalta, and J. Comellas, "Experimental Assessment of Bulk Path Restoration in Multi-layer Networks using PCE-based Global Concurrent Optimization," *Journal of Lightwave Technology*, vol. 32, pp. 81-90, 2014.

[24] Open Network Operating System (ONOS): [on-line] https://onosproject.org/.

[25] S. Alvarez, *QoS for IP/MPLS Networks*, 1st edition, Cisco Press, Jun. 2006.