Toward Multilayer Disaggregated Node Telemetry and Local Decision Making

Luis Velasco[1*], Lluís Gifre[2], and Jose-Luis Izquierdo-Zaragoza[1]

[1] *Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain*
[2] *Universidad Autónoma de Madrid (UAM), Madrid, Spain*
e-mail: lvelasco@ac.upc.edu

Abstract: A generic node agent supporting disaggregated node telemetry is presented. Data collection close to devices enable making local decisions, leveraging SDN controllers for network-wide operations. The agent is demonstrated in a BER-triggered transponder reconfiguration scenario.

OCIS codes: (060.4250) Networks; (060.1155) All-optical networks

1. Introduction

Node disaggregation (*whitebox*) focuses on radically reducing CAPEX without compromising network performance and/or features. 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 Infrastructure Element (IE) a self-contained element exposing some 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. Several initiatives are working on the definition of optical interoperability specifications and YANG data models, such as OpenROADM [1] and OpenConfig [2]. However, it is unclear how to operate disaggregated network and/or IT infrastructures at scale. Here, the role of node telemetry is gaining traction in order to provide real-time management capabilities to service providers and infrastructure owners. Node telemetry involves not only data gathering from devices, but also their (remote) control according to such data, either manually by a technician or automatically by some software procedure.

To this respect, authors in [3] define a distributed data analytics framework including *extended nodes*, which run close to the network nodes, and a big data centralized system, running in the *control and management* (C&M) *plane*. Extended nodes collect and collate monitoring data records from configured *observation points* (OP) in the nodes, which can be aggregated therein to reduce the amount of data sent toward the centralized system, and for Knowledge Discovery from Data (KDD) to proactively notify the centralized system about network anomalies and degradations. Nonetheless, the result from local KDD processes could be additionally exploited for local parameters reconfiguration in the network nodes, thus implementing local Observe-Analyze-Act (OAA) control loops, which could in turn trigger a number of network-wide OAA loops.

In this paper, we extend the architecture in [3] with a generic node agent for multilayer disaggregated whiteboxes, now supporting local OAA loops (i.e., modifying original configuration from the SDN controller). As an example of local OAA loop, we propose a use case to facilitate autonomic selection of the modulation format used in lightpaths (L0_LSPs) in response to variable transmission conditions; this can be done locally since modulation format can be easily detected by receivers [4]. Since changing modulation format of a lightpath might entail the corresponding capacity change of an L2 virtual link (vlink) in a multilayer scenario, traffic shaping functions could be also applied at both ends of the conveyed L2_LSPs (i.e., not locally) to increase or reduce the allowed traffic for such vlink. The proposed generic agent (*hypernode*) controls *components*, i.e., any configurable or monitorable element such as interfaces and LSPs, in the underlying IEs; an autodiscovery function allows retrieving the available monitoring capabilities per component, as well as already configured OPs. This novel functionality is of paramount importance in brown-field scenarios, where the network is already in operation.

2. Disaggregated data plane and monitoring architecture

An example of a multilayer partially disaggregated node is shown in Fig. 1; although for illustrative purposes, a very simple example is presented, the subsequent discussion can be easily generalized for more complex node architectures. In this example, the multilayer node consists of two IEs: one MPLS-TP switch and one optical node. Very briefly, the switch includes Ethernet interfaces, 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 vlinks. The MPLS-TP switch also integrates optical transponders, so vlinks are supported by lightpaths, i.e., L0_LSPs, in the optical layer. Ingress optical signals are forwarded to the collocated optical node, which routes them through the corresponding WDM interface toward a neighboring optical node. In this context, we define seven monitorable elements among the available components, each supporting a certain set of monitoring

The research leading to these results has received funding from the EC through the METRO-HAUL project (G.A. nº 761727), from the Spanish MINECO TWINS project (TEC2017-90097-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

https://doi.org/10.1364/OFC.2018.Th2A.28

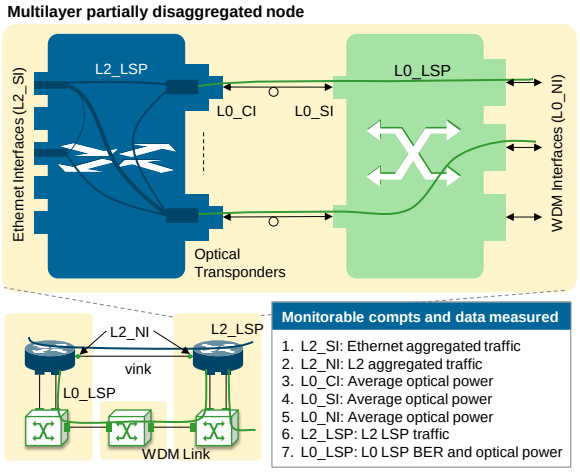

Fig. 1. Example of multilayer partially disaggregated node.

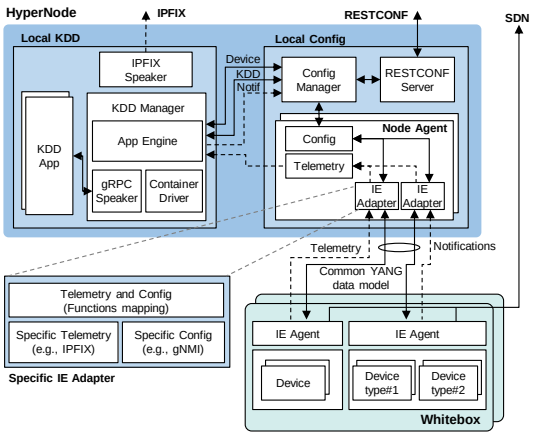

Fig. 2. Hypernode architecture.

capabilities through OPs: *i*) L2 server interface (L2_SI), monitoring incoming/outgoing Ethernet traffic; *ii*) L2 vlink endpoints (L2_NI), monitoring aggregated L2 traffic; *iii*) L0 transponder client interfaces (L0_CI), inside MPLS-TP switch, *iv*) L0 server interfaces (L0_SI), in the OXCs, and *v*) WDM interfaces (L0_NI), monitoring average optical power; *vi*) L2_LSPs, monitoring L2 traffic for each LSP; and *vii*) L0_LSPs, monitoring bit error rate (BER), power, and others, measured at the subcarrier module of the transponders.

The monitoring architecture proposed in Fig. 2 consists of a multi-node agent, referred to as *Hypernode*, designed to control and monitor one or more whiteboxes, where IEs in a whitebox might be controlled by different IE agents. Hypernodes augment extended nodes from [3] 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, through a RESTCONF-based North-Bound Interface (NBI), from the C&M plane. The NBI is based on a YANG data model that includes two subtrees: one for configuring the local KDD module inside the hypernode and another for configuring whiteboxes. A manager centralizes the configuration of the local KDD and underlying whiteboxes. Finally, a number of node agents (one per whitebox) are used to configure the whiteboxes and to collect monitoring data from them. Although IE agents' NBIs are based on a common YANG data model, different protocols for configuration and telemetry might be considered (e.g., gNMI for configuration and IPFIX for telemetry, or NETCONF for both of them). For this very reason, node agents include bespoke IE adapters implementing specific protocols and function mapping for the underlying IE agent.

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 C&M plane in two manners to apply network-wide OAA loops: *i)* through IPFIX messages including processed (i.e., averaged each 15 minutes) monitoring samples; or *ii)* through asynchronous notifications using the aforementioned RESTCONF NBI. In a hypernode, aiming to enable distributed node telemetry, local KDD can be additionally used for making decisions in local OAA loops and, therefore, KDD applications can issue configuration commands directly toward whiteboxes. In case of reconfiguration, hypernodes also notifies the C&M plane through the RESTCONF NBI.

3. YANG data model and operation

Fig. 3 presents the structure of the proposed YANG data model supported by all IEs defining: *i*) a configuration subtree including every programmable or monitorable component in the IE; and *ii*) a monitoring subtree to facilitate autodiscovery. A key element in the model is the *component* that represents any configurable or monitorable element in the underlying IEs; telemetry in monitorable components can be enabled and disabled by creating and deleting OPs. Note that monitorable components include a list of supported monitoring templates; they represent different measurement methods and, as such, each OP is associated to only one monitoring template. Although components under the configuration and monitoring subtrees are related, we relaxed 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 of such subtree. Whenever configuration of some component is required by a KDD application, the local configuration module retrieves it directly from the IE agent. After that, the C&M plane is notified about local changes both for data consistency purposes and triggering of subsequent network-wide OAA loops.

Fig. 4 presents the three workflows to be experimentally demonstrated in the next section. The autodiscovery workflow is initiated by the C&M plane, after a new node has been added to its topology database. Specifically, the *node-id* and the list of IE agents (each represented by two tuples, one for the configuration and another for the telemetry) are received in the hypernode to start the autodiscovery process. Upon the reception of the message, the local config block in the hypernode instantiates a new node agent with the defined IE adapters; each IE adapter connects to their peer IE agent to get the current monitoring subtree status. Finally, the complete

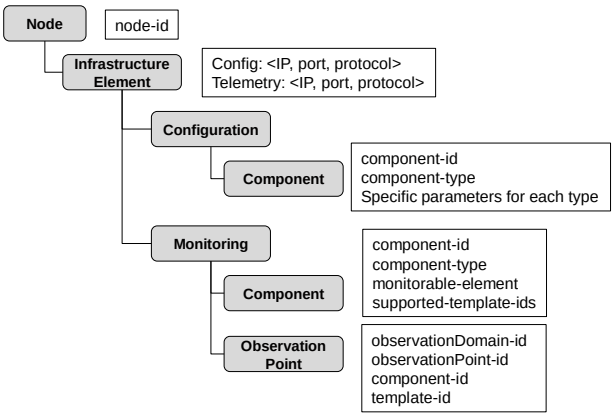

Fig. 3. Common YANG data model.

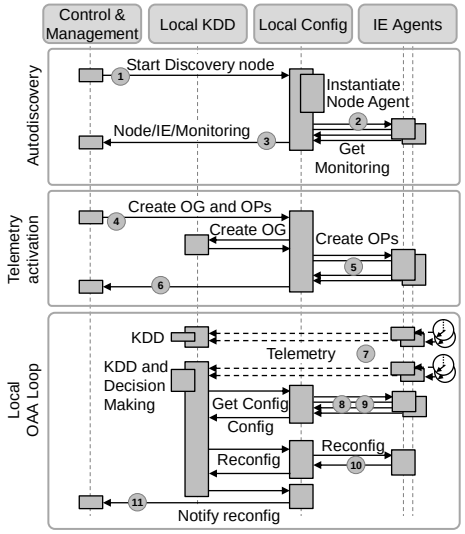

Fig. 4. Workflows demonstrated in this paper.

node's monitoring subtree is send back to the C&M plane.

The telemetry activation workflow is also initiated by the C&M plane: it issues a message to create one observation group (OG) in a KDD application running in the local KDD module to collect, analyze, and aggregate measures from a set of OPs to be activated in the components; once an OG has been created, add/remove OPs messages can be issued. Finally, the local OAA loop workflow runs after a data record has been gathered; monitoring records are analyzed by the corresponding KDD application and, when a certain pattern is detected, the configuration of some components may be changed. To that end, the KDD application requests the current configuration of a component to the local config module, which retrieves it from the IE agent. In our use case, upon detecting excessive BER in a L0_LSP, the corresponding IE can be interrogated about the currently configured modulation format, which can be immediately modified.

4. Experimental demonstration

The experimental demonstration has been carried out in the UPC's SYNERGY testbed. The scenario consists of a C&M plane system, one hypernode, and one multilayer node emulator, including two IE agents for an L2 switch and an OXC, all of them developed in Python. Fig. 5 presents a capture with the exchanged messages related to the workflows previously defined (message numbering is consistent with that in Fig. 4). Messages 1-3 and 4-6 represent autodiscovery and telemetry activation workflows, respectively, messages 7 are for telemetry gathering, whereas the rest cover the local OOA loop. Upon detection of BER-exceeded in an L0_LSP, the local KDD module asks the whitebox for the current configuration of such LSP (8). Next, the end transponder of the L0_LSP is interrogated to gather its supported modulation formats (9) and a new modulation format, among those supported by the transponder, is selected for the LSP (10). Finally, the C&M plane is notified (11).

	Source	Destination	Protocol	Info	
1	C&M	Hypernode	HTTP	POST /restconf/data/hypernode/nodes HTTP/1.1 (application/yang.data+json)	
2	Hypernode	Whitebox	HTTP2	SETTINGS, HEADERS, WINDOW_UPDATE, DATA	
	Whitebox	Hypernode	HTTP2	SETTINGS, HEADERS, DATA, HEADERS, PING, WINDOW_UPDATE	Autodiscovery
3	Hypernode	C&M	HTTP	HTTP/1.1 200 OK (application/yang.data+json)	
4	C&M	Hypernode	HTTP	POST /restconf/data/hypernode/applications[application-id=0]/observation-groups HTTP/1.1	
5	Hypernode	Whitebox	HTTP2	HEADERS, WINDOW_UPDATE, DATA	
	Whitebox	Hypernode	HTTP2	HEADERS, DATA, HEADERS	Telemetry activation
6	Hypernode	C&M	HTTP	HTTP/1.1 200 OK (application/yang.data+json)	
7	Whitebox	Hypernode	CFLOW	IPFIX flow (64 bytes) Obs-Domain-ID= 0 [Data:310]	
	Whitebox	Hypernode	CFLOW	IPFIX flow (64 bytes) Obs-Domain-ID= 0 [Data:310]	
8	Hypernode	Whitebox	HTTP2	HEADERS, WINDOW_UPDATE, DATA	
	Whitebox	Hypernode	HTTP2	HEADERS, DATA, HEADERS	
9	Hypernode	Whitebox	HTTP2	HEADERS, WINDOW_UPDATE, DATA	
	Whitebox	Hypernode	HTTP2	HEADERS, DATA, HEADERS	Local OAA Loop
10	Hypernode	Whitebox	HTTP2	HEADERS, WINDOW_UPDATE, DATA	
	Whitebox	Hypernode	HTTP2	HEADERS, DATA, HEADERS	
11	Hypernode	C&M	HTTP	HTTP/1.1 200 OK (text/event-stream)	

Fig. 5. Exchanged messages for the defined workflows.

5. Conclusions

In this paper, a generic node agent and a YANG data model were presented to enable distributed node telemetry, as well as local and network-wide OAA loops implementation. The agent has been experimentally demonstrated through a use case involving autodiscovery of components and monitoring capabilities, telemetry activation and local OAA for L0_LSP reconfiguration under signal degradation.

References

[1] OpenROADM: [on-line] www.openroadm.org
[2] OpenConfig: [on-line] www.openconfig.net
[3] L. Velasco *et al.*, "An Architecture to Support Autonomic Slice Networking," IEEE/OSA *JLT*, 2018.
[4] T. Bo *et al.*, "Modulation Format Recognition for Optical Signals Using Connected Component Analysis," *IEEE PTL*, 2017.