Monitoring and Data Analytics-triggered reconfiguration in partially disaggregated optical networks

Ll. Gifre[1*], F. Boitier[1], C. Delezoide[1], M. Ruiz[2], M. Buffa[3], A. Morea[3], R. Casellas[4], L. Velasco[2], P. Layec[1]

(1) Nokia Bell Labs, Nozay, France; (2) Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain; (3) Nokia, Vimercate, Italy; (4) Centre Tecnològic de Telecomunicacions Catalunya (CTTC), Castelldefels, Spain.

(*) e-mail address: lluis.gifre_renom@nokia-bell-labs.com

ABSTRACT

We present ADONIS, which stands for Aggregator/Disaggregator for Optical Network equIpmentS, a novel open device agent able to construct logical network devices from (dis)aggregation of physical components in order to expose meaningful network devices to the SDN controller. We experimentally assess it by means of a control closed-loop involving ADONIS, a Software Defined Network controller, a Monitoring and Data Analytics system, and a novel reconfiguration tool, SMART-A.

Keywords: Autonomic networking, Monitoring and data analytics, Software defined networks, Network disaggregation, Network reconfiguration.

1. INTRODUCTION

To build agile and highly reliable optical networks in a cost-efficient way, the telecom industry moved from fixed infrastructure defined at the "design phase" to flexible and reprogrammable hardware capable of supporting numerous operating cases. Software Defined Networks (SDN) controllers provide abstractions of network devices through open SouthBound Interfaces (SBI) enabling to centralize control and management tasks in a single system.

Furthermore, monitoring the state of network impairments and dynamically tuning components' configurations enable to reduce margins and entail a reduction in CAPEX [1]. This scenario demands, not only an SDN controller for reprogramming the devices, but also a specialized Monitoring and Data Analytics (MDA) system [2] to collate monitored data from the network components and to apply novel data analytics techniques, like Machine Learning, to extract knowledge and to take appropriate reconfiguration decisions. The network can be reconfigured in a proactive or reactive manner when undesired conditions are, respectively, predicted or detected. These automated control closed-loops also reduce OPEX and improve the network reliability and performance [3].

Besides, network disaggregation has been proposed in the last years aiming at improving the agility at minimal cost. To achieve this goal, control interfaces using open device models and protocols were standardized, thus enabling coordination between multi-vendor equipment such as in Open Line Systems [4],[5]. Nevertheless, when disaggregation level is high, each single component will expose its interface to the SDN controller leading to complex management of network elements, e.g. a *N*-degree ROADM will be a combination of multiple independent filter and amplifier cards. Similarly, the dual problem arises when a single shelf integrates components belonging to different network nodes, such as transponder and *N*-degree ROADM cards. However, up to now no generic device agent for (dis)aggregated network elements is available.

In this paper, *i*) we present ADONIS, introduced in [6], which stands for Aggregator/Disaggregator for Optical Network equIpmentS, a novel open device agent prototype, capable of logically disaggregating a single physical (commercial) device into multiple logical components and aggregating multiple physical (commercial) devices into a single logical component, and exposing the resulting logical devices to the SDN controller; *ii*) we also present a workflow implementing a simple closed-loop that triggers the rerouting of a set of lightpaths when a soft-failure affecting them, such as a degradation, is detected; *iii*) we integrate ADONIS with novel SMART-A reconfiguration tool, ONOS SDN controller [7], CASTOR MDA system [2] to provide the required monitoring and reconfiguration capabilities to the control and management plane; and *iv*) we experimentally assess ADONIS together with the rest of components in a 4-node meshed optical network testbed with ROADMs from 2 different vendors providing experimental results describing the messages exchanged and elapsed time in the operations.

2. ARCHITECTURE

The architecture of our testbed network control and management plane is depicted in Figure 1a. The ADONIS open device agent logically (dis)aggregates the physical devices into logical devices and interfaces them with ONOS. ADONIS is implemented in Python by Nokia Bell Labs. The NorthBound Interface (NBI) relies on NETCONF/YANG offering OpenROADM [8] interfaces for ROADMs and OpenConfig [9] ones for transponders. We illustrate the use of ADONIS in Figure 1b showing the physical (commercial) data plane and in Figure 1c depicting the logical devices composing the 4-node network topology.

The physical data plane consists of one Nokia 1830 PSI-2T 4-lineport transponder (Figure 1b, upper right) disaggregated by ADONIS into 4 independent logical transponders (labelled TP1 to TP4 in Figure 1c); two Lumentum RDM20 degree boxes (Figure 1b, upper right) are aggregated into one virtual 2-degree ROADM

(labelled R4); and one Nokia 1830 PSS-32 shelf (Figure 1b, bottom left), holding 7 iROADM9R cards, is aggregated into 3 logical 2-degree ROADMs (labelled R1, R2 & R3), 1 card is not used. For R1, R2 and R3, the ADONIS first needs to disaggregate the shelf into its individual cards and then aggregate them back into logical ROADMs.

Variable Optical Attenuators (VOA) are deployed in both directions of links R1-R2 and R1-R3 to emulate optical fiber perturbations. Moreover, to emulate additional lightpaths and bring stability on optical power in each network port, the testbed is pre-loaded with some loading channels also controlled by the SDN controller thanks to emulation functionalities that we implemented in ADONIS. From the physical point of view, they are implemented by means of an optical amplifier chained with optical filters and splitters. By controlling add-drop ports in ROADM degree cards, we route the loading channels through the network as we do for real channels. As a result, the real network seen by the SDN controller has 38 logical nodes (4 logical real ROADMS, 4 logical real transponders, and 10 logical emulated transponders in each of R1, R2 and R3), and 76 unidirectional optical links. For the sake of clarity, emulated components have not been depicted in Figure 1b,c.

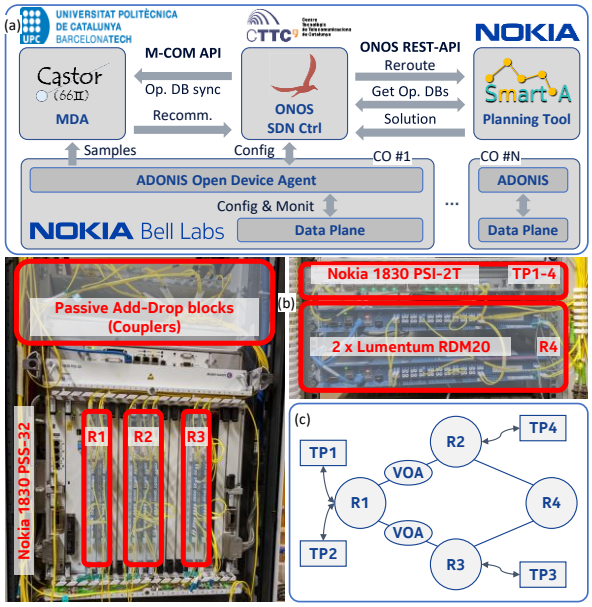

Figure 1. Proposed architecture: (a) software components of the network controller, (b) physical equipment and (c) logical devices and network.

The CASTOR MDA system [2] collects and processes monitored data from the network equipment and issues appropriate recommendations to the SDN controller. CASTOR is implemented in Python by UPC following a micro-services architecture for the backend. The MDA is extended with an application implementing a degradation detection algorithm that triggers the control closed-loop presented in section 3. The M-COM interface [10] is used to populate the MDA operational databases, in particular, the Traffic Engineering Database (TED) and the Label Switching Path Database (LSPDB), and to issue recommendations to the SDN controller. To build a monitoring system scalable and performant, given the heterogeneity of network equipment and the volume of monitored samples, we choose to use the standardized IPFIX protocol [11] to convey data from ADONIS to CASTOR. IPFIX was initially designed to convey packet layer-related monitoring data; we defined some custom fields and templates to extend it and enable conveying optical layer devices states, such as transmitted and received power, as we detail in section 4.

The ONOS SDN controller [7] manages the logical devices composing the network topology. We use ONOS v2.3.0 with the Open Disaggregated Transport Network (ODTN) drivers which provides the NETCONF/YANG–based SBIs used by ADONIS. Two additional applications have been deployed, the first one implements the M-COM interface enabling the coordination between CASTOR and ONOS, while the second one enables translating and then forwarding CASTOR recommendations into SMART-A computation requests.

The SMART-A reconfiguration tool is developed by Nokia Italy in C++. It is responsible for solving the optimization problems requested by ONOS, optionally asking the network operator for validation, and implementing the computed solution in the network through ONOS. It integrates a REST API to receive computation requests from ONOS and uses ONOS standard NBI to retrieve up-to-date information available regarding devices, links and connections, and to update connections after a solution is validated.

3. WORKFLOWS

Three workflows, reproduced in Figure 2, are used in this work. The first workflow (a), is manually triggered by the network operator when the network controller is initialized. It consists in an initial synchronization of the operational databases, i.e. TED and LSPDB, belonging to ONOS and CASTOR. Then, ONOS discovers logical network devices; for each device, it retrieves its details by means of ADONIS and issues an operational databases update notification to CASTOR. The second workflow (b), is manually triggered by the network operator every time a new connection is created in ONOS; it configures the appropriate rules in the logical devices along connection's path and issues appropriate update notifications to CASTOR to synchronize its operational databases.

The third workflow (c) implements the control closed-loop. Periodically, ADONIS retrieves monitoring data samples from the physical devices, associates them to the corresponding logical devices and forwards collected data to CASTOR by means of IPFIX messages (label 1 in Figure 2c); CASTOR stores the samples on its internal repository and forwards them to the degradation detection algorithm (2). The algorithm correlates the samples with the network components and analyses trends to identify and localize the origin of degradations by comparing

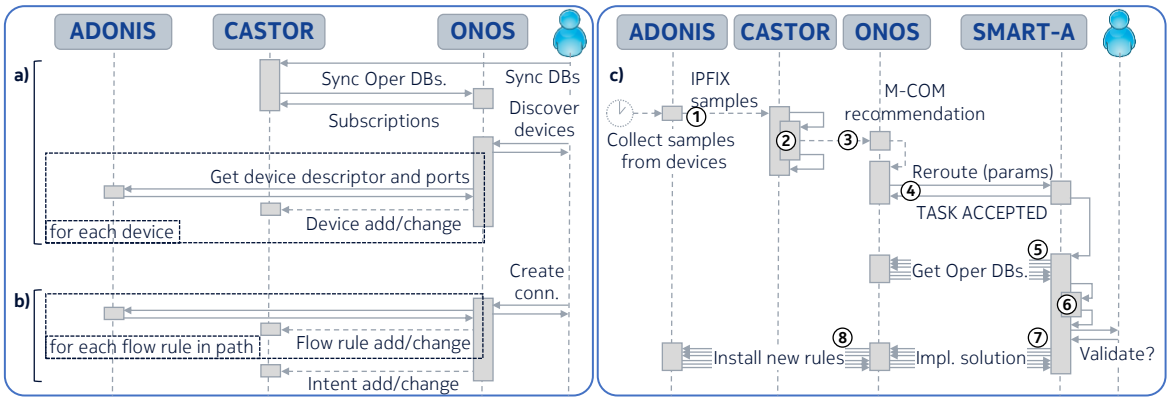

Figure 2. Proposed workflows: (a) initial set-up, (b) per-connection creation, and (c) closed-loop.

measurements on transmitted and received power. In case of detecting a degradation, the algorithm issues a reconfiguration recommendation to ONOS (3) including the component(s) to be avoided and the affected optical connections. When ONOS receives the recommendation, it forwards a request to SMART-A with similar content (4); the latter, retrieves from ONOS an up-to-date copy of the operational databases (5) and solves the optimization problem to reroute the affected connections taking into account optical layer constraints and aiming at minimizing their reconfiguration downtime (6). When SMART-A finds a solution, it optionally asks for validation to the network operator, and issues requests to ONOS to implement it (7). ONOS then compiles the new optical connection intents and configures the appropriate rules in the logical devices through ADONIS (8).

4. EXPERIMENTAL VALIDATION

We carried out the experimental validation in Nokia Bell Labs France testbed deploying the architecture, workflows and applications, and configured the logical (dis)aggregations described in sections 2 and 3. Next, we initialized the scenario for both real and emulated optical connections following the initialization workflows. We configured 2 real bidirectional optical connections, illustrated in Figure 3a, in the test-bed, plus 10 loading channels between emulated transponders. Next, we increased in 5 dB the attenuation of VOAs in link R1-R2 to trigger the reconfiguration workflow.

In Figure 3c, we list the Wireshark capture containing the relevant messages conveniently labelled according to the reconfiguration workflow. Periodically, samples retrieved by ADONIS are forwarded to CASTOR by means of IPFIX messages (label 1 in Figure 3c); CASTOR processes the samples by running the degradation detection algorithm (2) and, as a result of detecting a degradation in link ROADM1-ROADM2, the MDA sends a message to ONOS through the M-COM interface (3) recommending to avoid that link. Upon reception of the message, ONOS forwards a request with similar content to SMART-A to reroute the affected connections (4). Next, SMART-A retrieves the operational databases from ONOS (5) and solves the optimization problem (6) and implements the solution found (7). Eventually, ONOS installs the appropriate rules in each logical device supported by ADONIS (8). The resulting paths are illustrated in Figure 3b.

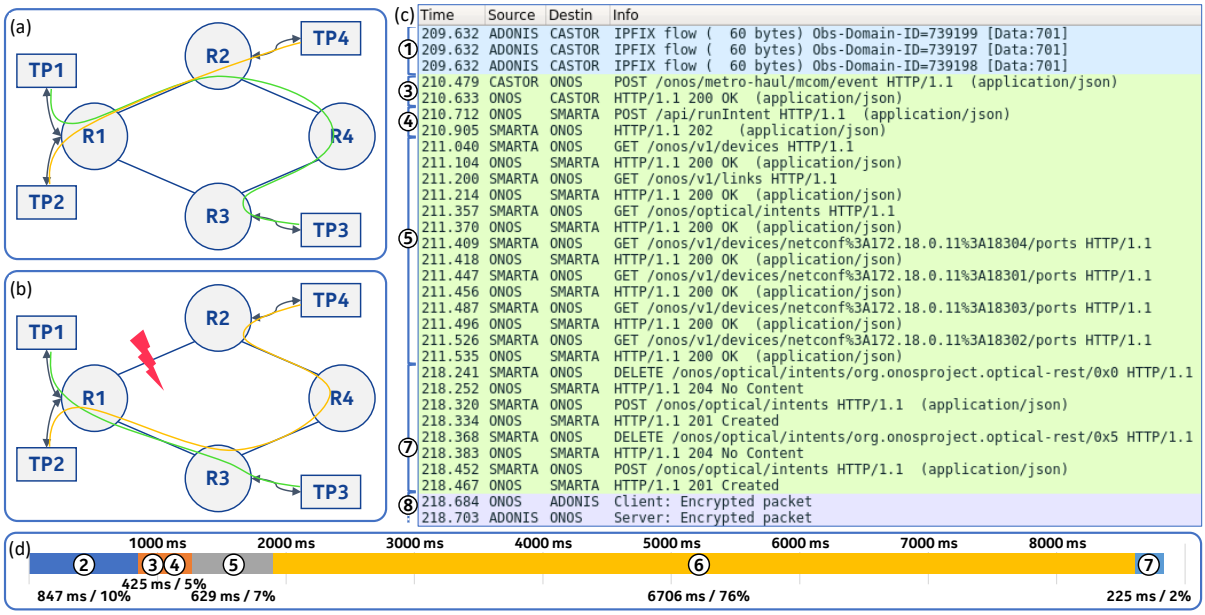

Figure 3. Experimental validation: (a) connections in normal condition, (b) connections after rerouting, (c) Wireshark capture with relevant messages exchanged, and (d) time consumed by the workflow in each step.

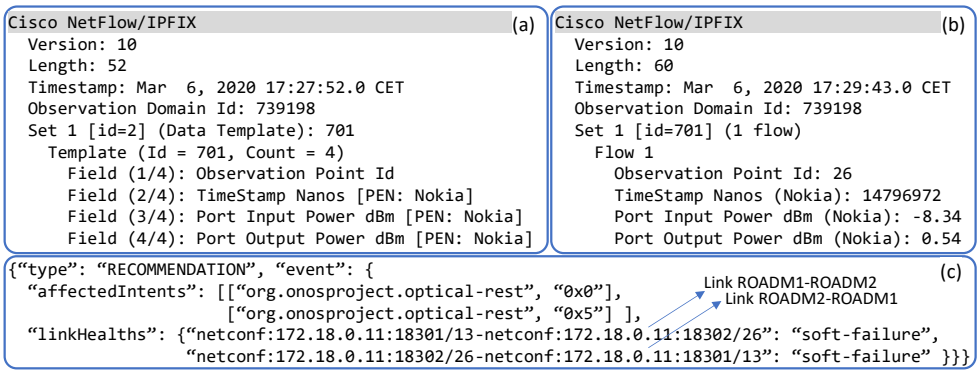

Figure 4. Relevant messages exchanged: (a) IPFIX template record, (b) IPFIX data record (message 1), and (c) JSON recommendation from MDA to ONOS (message 3).

Figure 3d shows the workflow timeline of the execution. The complete workflow took around 9 seconds; mainly driven by the planning tool. Note that a 38-node 76-link is retrieved by SMART-A (real and emulated logical hardware) and used as input to solve the optimization problem. The operations performed in that time frame includes construction of internal data structures and validation of topology, compute the optimal routing solution for the paths, and validate the solutions. The time contributed by the MDA was 847 ms, including the storage of samples on its internal repository and execution of the degradation detection algorithm.

The content of IPFIX and recommendation messages are illustrated in Figure 4. Figure 4a details the custom IPFIX template record message (id=701); it contains 4 fields: *i*) standard field "`Observation Point Id`" used to encode the device port identifier, *ii*) custom 32-bit unsigned integer field "`TimeStamp Nanos`" carries the fractional part of the timestamp to improve accuracy of IPFIX header's timestamp field from seconds to nanoseconds, and two custom 32-bit floating point fields *iii*) "`Port Input Power dBm`" and *iv*) "`Port Input Power dBm`" carrying, respectively, the input and output optical power of ROADM degree line ports. Figure 4b depicts an example of IPFIX message containing one sample belonging to port 26 in logical device with unique identifier 739198. The recommendation is shown in Figure 4c; field "`affectedIntents`" carries the list of connections affected by the degradation, while "`linkHealths`" contains the labels for links where the degradation was localized. Labels consist in a health state computed by the algorithm. In particular, "`normal`" links (default if unlabelled) are safe to be used, "`hard-failure`" links are suffering from significant degradation so they are strongly discouraged and, the MDA recommends to avoid "`soft-failure`" links, when possible, since they have traces of degradation.

5. CONCLUSIONS

In this paper, we presented ADONIS, a novel open device agent prototype, able to logically dis(aggregate) and emulate physical network equipment and exposing their logical abstractions to the SDN controller. We presented a reconfiguration workflow involving ADONIS, ONOS SDN controller, SMART-A reconfiguration tool and CASTOR MDA system, and experimentally assessed them in a multi-vendor 4-node meshed optical network testbed increased with logically emulated hardware.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the EC through the METRO-HAUL (G.A. nº 761727).

REFERENCES

[1] S. Oda, et al., "A Learning Living Network With Open ROADMs", IEEE/OSA JLT, vol. 35, pp. 1350-1356, 2017.
[2] Ll. Gifre et al., "Autonomic Disaggregated Multilayer Networking", IEEE/OSA JOCN, vol. 10, pp. 482-492, 2018.
[3] D. Rafique, et al., "Cognitive Assurance Architecture for Optical Network Fault Management", IEEE/OSA JLT, vol. 36, pp. 1443-1450, 2017.
[4] N. Sambo, et al., "Experimental Demonstration of a Fully Disaggregated and Automated White Box Comprised of Different Types of Transponders and Monitors", IEEE/OSA JLT, vol. 37, pp. 824-830, 2018.
[5] F. Paolucci, et al., "OpenConfig Control of 100G/400G Filterless Metro Networks with configurable Modulation Format and FEC", in Proc. IEEE/OSA Optical Fiber Communication Conference (OFC), 2019.
[6] Ll. Gifre, et al., "Demonstration of Monitoring and Data Analytics-triggered reconfiguration in partially disaggregated optical networks", in Proc. IEEE/OSA Optical Fiber Communication Conference (OFC), 2020.
[7] Open-source Network Operating System (ONOS) project [online] available 2019. https://onosproject.org/
[8] OpenROADM MSA [online] available 2019. http://www.openroadm.org/
[9] OpenConfig [online] available 2019. http://openconfig.net/
[10] L. Velasco et al., "Building Autonomic Optical Whitebox-Based Networks", IEEE/OSA JLT, vol. 36, pp. 3097-3104, 2018.
[11] B. Claise, B. Trammell, P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", IETF RFC 7011, September 2013.