Experimental Assessment of Node and Control Architectures to Support the Observe-Analyze-Act Loop

Lluís Gifre[1], Alba P. Vela[1], Marc Ruiz[1], Jorge E. López de Vergara[2], and Luis Velasco[1*]

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

Abstract: An architecture supporting the OAA loop is proposed. It consists on extending nodes and the domain controller with analytics capabilities for local and network-wide operation automation. The architecture is experimentally assessed through a use case.

OCIS codes: (060.4250) Networks; (060.4256) Networks, network optimization

1. Introduction

Data analytics is an interesting and challenging task to find patterns in heterogeneous data coming from different sources. In networking, data comes from monitoring the data plane, e.g., received power, impairments and errors in the optical layer or service traffic in the MPLS layer. The output of data analytics can be used for automating network operation [1], detecting traffic anomalies [2], or transmission degradation [3]. This way of doing network management is collectively known as the *observe-analyze-act* (OAA) loop since it links together monitoring, data analytics and operation (with or without re-optimization).

In this paper, we propose an architecture to support the OAA loop in optical networks. First, the network nodes are extended with data analytics capabilities to perform local Knowledge Discovery from Data (KDD) from monitoring data and make local decisions thus, reducing both, the amount of data to be collated at the control/management plane and the reaction time [2], [4]. Second, the Software-defined Networking (SDN) controller is extended with a domain analytics block to collect monitoring data from the network nodes, implement KDD and make domain-wide decisions. The proposed architecture is experimentally assessed through a use case to adapt the capacity of Label Switched Paths (LSP) served on a MPLS-over-optical virtual network to follow traffic estimations aiming at reducing resource utilization.

2. Proposed Architecture

The proposed architecture to support the OAA loop is presented in Fig. 1 that includes: the set of extended nodes (e.g., MPLS switches, optical nodes, etc.) and the extended domain controller.

In the nodes, data analytics functionalities have been added to implement KDD and make decisions on the monitored data. A local temporal data storage is used to aggregate monitoring data: monitoring data can be obtained from the devices using a fine period (e.g., 1 min) and immediately be available for local KDD algorithms, so as to provide near real-time reaction against errors or anomalies in the data plane [4]. However, aiming at reducing the amount of data to be collected and stored in the centralized domain controller, monitoring data is not immediately sent to the domain controller; instead, it is aggregated and conveyed to the domain controller using a programmable, usually coarser, period (e.g., 15 min). Monitoring data is reported to the domain controller by means of the IPFIX protocol [5]. One of the interesting features of IPFIX is that it supports templates with custom fields, which increases the flexibility of the protocol that can be used in different contexts, such as the optical layer to monitor specific optical parameters, e.g., impairments, received power, bit error rate, etc., the MPLS layer to monitor per-LSP traffic, etc. Table 1 presents the proposed template to monitor LSPs' bitrate. Among the defined fields, the template includes custom fields for the *Symbolic Path Name* of the monitored LSP and the average bitrate in the monitored period.

Regarding monitoring parameters configuration, we have developed a YANG model extending the generic model for connection oriented OAM protocols defined in [6]. A RESTCONF [7] interface is used for the domain controller to configure such parameters in the nodes. Among the parameters that can be configured in a MPLS switch, we can mention: the monitoring period for monitoring data aggregation, Symbolic Path Name of the LSPs to be monitored (the Symbolic Path Name of the LSPs are specified in the nodes at setting-up time), model coefficients for LSP traffic estimation, and threshold values for traffic anomaly detection [4].

In the domain controller, an IPFIX speaker collects aggregated monitoring data from the nodes and stores them in a big data repository. Periodically, collected data is processed and transformed into modeled data using data stream mining sketches. Algorithms for KDD are applied to modeled data to find patterns. For instance, in an MPLS network scenario, a model for each LSP traffic can be fitted and used to estimate future traffic conditions. A Decision Maker (DM) module can be used to make decisions based on predicted values, e.g., it can decide to modify the capacity allocated for an LSP in case that the traffic is predicted to drop. The DM

The research leading to these results has received funding from the Spanish MINECO SYNERGY project (TEC2014-59995-R), from MINECO/FEDER TRÁFICA project (TEC2015-69417-C2-1-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

Table 1. Proposed IPFIX template for LSP traffic monitoring

Field Name	Description	Units
`observationDomainId`	Id of the exporting device	---
`Timestamp`	Timestamp of the sample	UTC since epoch
`observationPointId`	Id of an observation point used to limit the scope of the monitoring. We use it to carry the Symbolic Path Name of the monitored LSPs.	---
`packetDeltaCount`	Number of packets since the previous report (if any) for the LSP.	packets
`layer2Bitrate`	Average L2 bitrate since the previous report (if any) for the LSP.	b/s

module generates data that can be used by an optimization module (planning tool) to produce an optimal network configuration. Finally, a RESTCONF interface is used to configure monitoring parameters in the nodes and receive asynchronous notifications from them (e.g., after a threshold value has been exceeded).

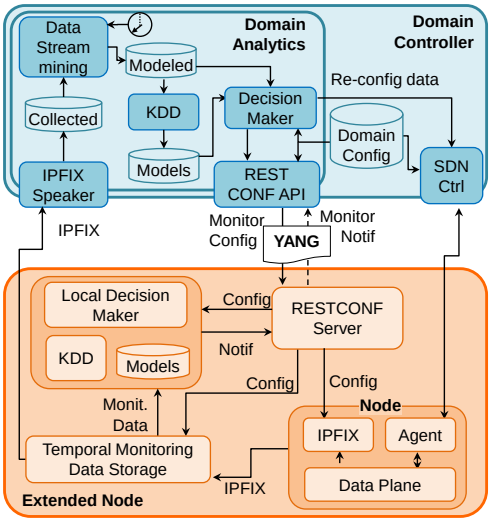

Fig. 1. Engine Architecture

3. OAA loop use case: dynamic LSP capacity adaptation

To illustrate how the proposed architecture can be used to implement the OAA loop, in this section we present a use case where the capacity of LSP is modified to adapt that reserved capacity to future traffic prediction. Fig. 2 presents a scenario where two LSPs are using a common virtual link between nodes S2 and S3. The graph in Fig. 2 plots the traffic model for the two LSPs and the aggregated load in the common link, where the total load is under 90Gb/s at any time of the day. If we manage the capacity of the LSPs to follow the traffic prediction, the capacity of the virtual link can remain unchanged to 100Gb/s. On the contrary, if the capacity of the LSPs remain constant and set to the maximum along the day (55 and 75Gb/s), the capacity of the virtual link needs to be increased to 200Gb/s by setting-up a parallel lightpath in the optical layer thus, consuming extra transponders.

To implement the LSP capacity adaptation use case, we propose the workflows presented in Fig. 3. The first workflow (a) covers the process of collecting monitoring data from the network nodes and transforming it into modeled data. Periodically, the network nodes send traffic monitoring data using IPFIX to the domain analytics (message 1 in Fig. 3a); these data in conveniently stored in the collected data repository for the specific LSP and the received timestamp. With a coarser period, a data stream mining sketch gets the last collected data (message 2) and transforms them into modeled data (message 3); the simplest data stream mining sketch computes min, max and average values but more complex algorithms can be implemented. The second workflow (b) focuses on computing a traffic model for every monitored LSP. The KDD process can be in charge of creating models by fitting polynomials, using artificial neural network (ANN) or any other data mining algorithm. To that end, modeled data is first retrieved from the repository (message 4 in Fig. 3b) and, once the traffic model has been created, it is stored in a traffic model repository (message 5). Finally, the last workflow (c) targets at managing the capacity reserved for each LSP as a function of the estimated traffic for the near future. The DM module periodically retrieves the model for every LSP (message 6 in Fig. 3c) and predicts the next traffic conditions. The prediction is then used to request a LSP capacity reconfiguration to the SDN controller (message 7), which applies the recommended capacity modification in the network (message 8).

4. Experimental assessment

The experimental assessment was carried out in our SYNERGY test-bed. The domain controller was implemented in Python; in addition to the modules described in section 2, a web application containing the

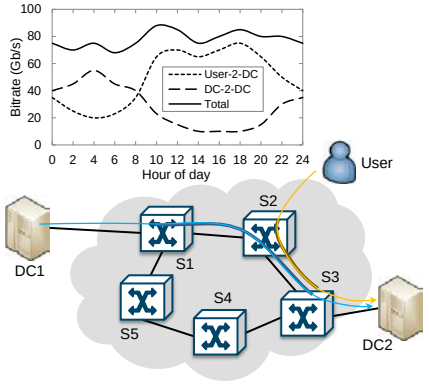

Fig. 2. Traffic Model-based LSP Capacity Adaptation

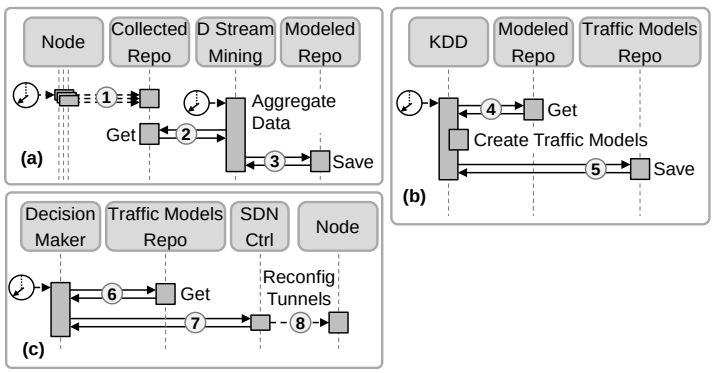

Fig. 3. Proposed workflows

domain database was developed to control the rest of modules. Cassandra v3.7 was deployed as the data analytics repository and its native protocol (named as Cassandra Query Language (CQL)) was used to interact with the database engine. The data analytics engine was implemented in Spark v2.0; a REST API server was implemented to enable receiving computation requests (from the domain controller). The data plane was deployed using Mininet v2.2 to build the multi-layer MPLS network topology, where nodes were based on OpenVSwitch (OVS) v2.5 configured to issue per-flow IPFIX monitoring data to the collectors at the extended nodes. The SDN controller was based on Ryu v4.6 and relied on OpenFlow 1.3 and OVS database protocol (OVSDB) to configure the flows and internal queues in the nodes, respectively. A REST API interface was implemented to enable LSP reconfiguration requests from the domain controller.

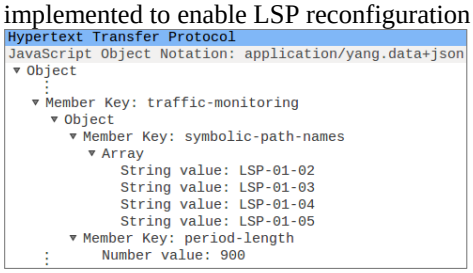

Fig. 4. Node monitoring configuration detail

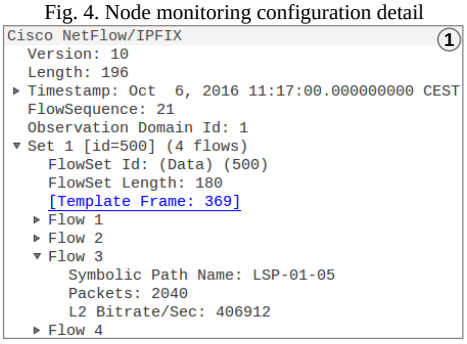

Fig. 5. IPFIX message detail

Fig. 4 presents the initial configuration of the `traffic-monitoring` YANG object for a node; in particular the monitoring period is set to 15 minutes and monitoring points for a set of LSPs identified by their Symbolic Path Names are activated.

Monitoring data in IPFIX messages (message 1 in the workflow in Fig. 3a) include template Id 500 (Table 1) that contains, among others, the number of packets and Layer 2 bitrate monitored at the monitoring points (Fig. 5). Note that data related to multiple monitoring points can be reported in a single IPFIX message provided that a single template is used. After receiving the IPFIX message, the domain controller stores the reported monitoring data in the collected data repository using CQL.

Periodically, a data stream mining process retrieves collected data using a CQL SELECT query (message 2 in Fig. 6 and Fig. 3a) specifying the monitoring point and the time range (e.g., the last hour). A CQL INSERT query (message 3 in Fig. 6 and Fig. 3a) stores modelled data in the repository after it has been processed.

The KDD module uses modeled data to create predictive traffic models that are used by the DM module; in our use case, DM updates the bitrate of the monitored LSPs. Fig. 7a presents the sequence of messages to implement the workflow in Fig. 3c. First, DM issues a REST API request (Fig. 7b) to SDNCtrl specifying the domain Id the LSP belongs to (0 in the example) and the LSP's symbolic path name. In the body, the message contains a JSON-encoded message containing the operation to be performed and the required parameters. To modify LSP's capacity, we use the traffic shaping functionality in OVS.

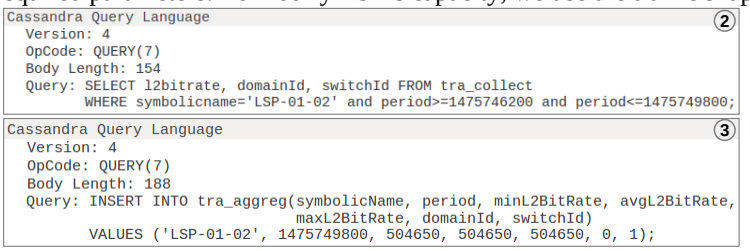

Fig. 6. Data Stream Mining processing

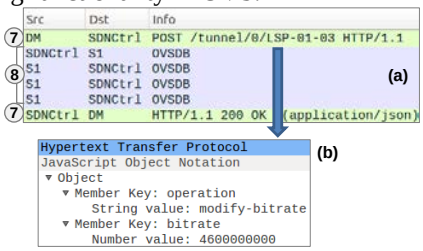

Fig. 7. LSP Bandwidth Update

5. Conclusions

An architecture to support the OAA loop that extends the network nodes and the domain controller has been proposed for optical networks. In the network nodes, data analytics capabilities that can be used to detect patterns (e.g., degradations) and make local decisions have been added. In the extended domain controller, data repositories are used to store monitoring data from the network nodes. A data stream mining process transforms collected into modelled data that is analyzed by the KDD process to find patterns. A DM module is used to make decisions and generate output data that the SDN controller can use for re-configuration purposes.

The proposed architecture has been experimentally assessed in our SYNERGY test-bed through an OAA loop use case to dynamically reconfigure the bitrate of LSPs.

References

[1] F. Morales et al., "Virtual Network Topology Adaptability based on Data Analytics for Traffic Prediction," JOCN, 2016.
[2] A. P. Vela et al., "Bringing Data Analytics to the Network Nodes," in proc. ECOC, 2016.
[3] M. Ruiz et al, "Service-triggered failure identification/localization through monitoring of multiple parameters," in proc. ECOC, 2016.
[4] A. P. Vela et al., "Reducing Virtual Network Reconfiguration and Traffic Losses under Multiple Traffic Anomalies," in ACP, 2016.
[5] B. Claise et al., "Specification of the IPFIX Protocol," IETF RFC 7011, 2013.
[6] D. Kumar et al., "Generic YANG Data Model for Connection Oriented OAM protocols", IETF draft, work in progress, 2016.
[7] A. Bierman et al., "RESTCONF Protocol," IETF draft, work in progress, 2016.