Interfaces for Monitoring and Data Analytics systems

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

ABSTRACT
Software defined networks (SDN) represents one of the most relevant innovations in recent years for the telecom industry. Major operators are planning to progressively migrate their transport networks, including optical, to such paradigm to take advantage of the programmability of connectivity services. In SDN, resources are abstracted at the networking level and exposed via a north-bound interface to management systems. On the other hand, Monitoring and Data Analytics (MDA) systems can collect monitoring data and make them accesible to external systems, such as billing platforms, to accurately charge users for the services they consume. In addition, in dynamic scenarios, where resources could be created, modified and removed on demand, MDA systems can also facilitate autonomic networking, and thus coordination between SDN and MDA is strictly required. In this paper, we present two monitoring interfaces for MDA systems: the Monitoring (M)-Control, Orchestration, and Management (COM) interface, an east-west interface that enables MDA systems to gather and synchronize operational and telemetry information from multiple COM modules, e.g., an SDN controller or a cloud resource manager, and issue recommendations to the COM modules as a result of data analysis; and the IO4 interface, a MDA north-bound interface that facilitates exporting monitored data to external systems. Functionality of each interface is described from the conceptual viewpoint and relevant operational workflows are proposed.
Keywords: Autonomic networking, Cognitive networking, Monitoring and data analytics

1 INTRODUCTION

With the advent of Software Defined Networks (SDN) and the stringent constraints demanded by new 5G-based applications and services, network automation became primordial. New applications such as remote surgery, vehicle driving engines or virtual/augmented reality require, not only a higher bitrate and throughput, but also stringently reduced latencies, jitter and packet loss. In that regard, the only possibility available for network operators to fulfill those requirements is to deploy autonomous networking platforms that keep track of the state of the network resources and, when possible, reconfigure the network in a proactive manner to minimize service disruptions.

Network automation [1], in fact, requires of an agile interaction between the Control, Orchestration, and Management (COM) systems [2], e.g., SDN Controllers, and those Monitoring and Data Anlytics (MDA) platforms in charge of collecting and analysing network equipment state. In that regard and aiming to enable correlation between the network devices and the monitoring data collected from them, the MDA platform must have access to the operational databases in the COM, e.g., the Traffic Enginering Database (TED) and the Label Switched Path Database (LSP-DB) containing, respectively, the network topology and the connections established in the network. In addition, a new database, named as Monitoring Database (MDB), needs to be defined in the MDA to manage the locations originating monitoring data, e.g., the Observation Points (OP), in the network. A network OP can be defined as the location in a network equipment where a sample containing some kind of device state information ins taken. An additional requirement for network automation includes bringing access to Operation/Business Support Systems (OSS / BSS) to the monitored data for different purposes, such as traffic forecasting, billing, etc. Given the complexity of the network, it makes sense to define a new simplified interface to facilitate access to monitoring data.

In this paper, we present the M-COM and IO4 monitoring interfaces and their integration in a distributed MDA platform like CASTOR [3], to provide means for synchronizing the operational databases of related COM modules, issue recommendations to those COM modules based on monitored data, and collect/configure telemetry for those network/IT devices with that capability. Note that this concepts can be extended to any network/IT equipment; however, in this paper, we will insight in network equipment.

2 MONITORING AND DATA ANALYTICS ARCHITECTURE

Despite of the fact that internal details of COM modules are, in general, hidden by interfaces, it is important to understand the internals of the monitoring module to help separation of functionalities between M-COM and IO4 interfaces. Figure 1 overviews a simplified view of the architecture of the MDA controller of CASTOR [1], a distributed MDA platform, extended with the M-COM and IO4 monitoring interfaces. CASTOR is composed of different modules according to a monitoring hierarchy with different levels of visibility, i.e., node-wide and network-wide. To this respect, we only consider the centralized network-wide module, which is actually the one interfacing both M-COM and IO4. From Figure 1, we can clearly define the expected functionalities supported by each interface.

The MDA controler offers two South Bound Interfaces (SBI) to interact with MDA Agents, i.e., monitoring nodes placed in a Central Office (CO) to collect, aggregate and pre-analyze samples from multiple, possibly disagregated, network devices and forward processed data to the MDA controller. When data (IPFIX Samples or RESTConf notifications) is received at the MDA controller, the *Data Manager* block is in charge of storing them in the *Collected Data Repository*, a scalable multi-master database such as Apache Cassandra, through the *Data Collector API*. The *KDD Manager* block serves as a container for running user-defined KDD processes in charge of processing incoming samples and notifications. To do so, KDD processes can subscribe to triggering events issued by the *Data Collector API* upon the reception of new samples and notifications. To support operation of the KDD processes, an *Operational Databases* block has been included. This block stores a clone of the operational databases in the connected COM modules, e.g., the network topology and connections databases of an SDN controller or the Virtual Network Functions (VNFs) and service chains in a Network Function Virtualization (NFV) orchestrator.

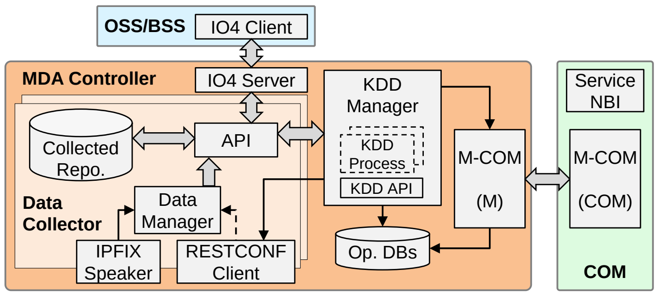

Figure 1. Architectural diagram

Additional monitoring interfaces have been defined in the context of network/IT telemetry, enabling not only data monitoring itself but also simple remote configuration of the data collection, collation and correlation processes, as well as possible decision-making or, at least, feedback to the COM modules. The M-COM and IO4 interfaces abstract a set of functionalities related to these tasks. Specifically, to facilitate the interaction between the MDA and COM systems, the M-COM interface has been defined to provide means for synchronizing the Operational Databases from the COM to the MDA controller, issuing recommendations for network/IT reconfiguration from the MDA controller and, optionally, receiving monitoring samples directly from the COM module. Similarly, the IO4 interface has been defined to enable external systems, such as an OSS/BSS, to retrieve monitoring samples from the MDA controller for different purposes that could include, amobg others, billing and service forecasting.

3 M-COM INTERFACE

The M-COM interface is devoted to synchronize information from operational databases from the COM module to the monitoring platform. This information can be correlated with monitoring data and enable issuing notifications to the COM module about different events, even providing recommended actions. The M-COM interface can also be used to manage telemetry functionalities from monitorable network/IT devices; in that regard, the M-COM offers methods to create/modify/delete OPs referring to telemetry streams in network/IT devices thus, configuring them to export telemetry data to the appropriate MDA platform components.

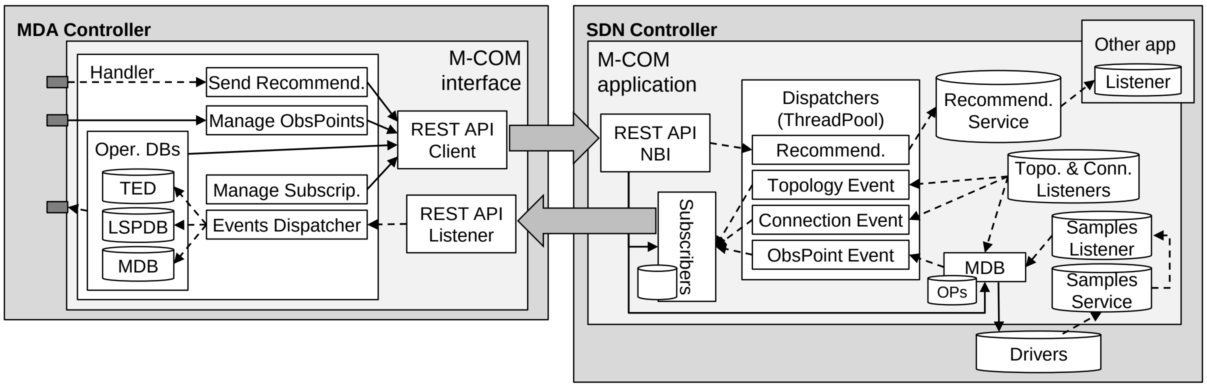

Figure 2. Components of the M-COM interface

Figure 2 illustrates the components belonging to the M-COM interface at both, the MDA controller and SDN controller sides. At the COM module side, the M-COM interface is implemented as a pluggable application and consists of: *i*) the *REST API NBI* block publishes M-COM management commands on the COM North Bound Interface (NBI) to receive control commands, *ii*) the *Subscribers* block keeps track of the subscribers that requested to receive database updates from the COM module and distributes events to them, *iii*) the *Dispatchers* block is implemented as a thread pool and integrates the logic to process M-COM – related events, *iv*) the *Recommendation Service* provides a means for other applications in the SDN controller to subscribe to messages received from the MDA controller, *v*) the *Topology & Connection Listeners* are in charge of capturing events from the COM operational databases and triggering the execution of appropriate tasks in the *Dispatcher* block, *vi*) the *MDB* block keeps track of the OPs requested by the MDA controller and configures the COM *Drivers* to enable telemetry and send the collected samples directly to the MDA controller, *vii*) the *Samples Service* enables *Drivers* to push their telemetry data to be received by those Sample Listeners connected to the service, and *viii*) the *Samples Listener* collects the telemetry samples issued by the COM *Drivers* and, if configured, forwards them to the subscribers of the MDB database.

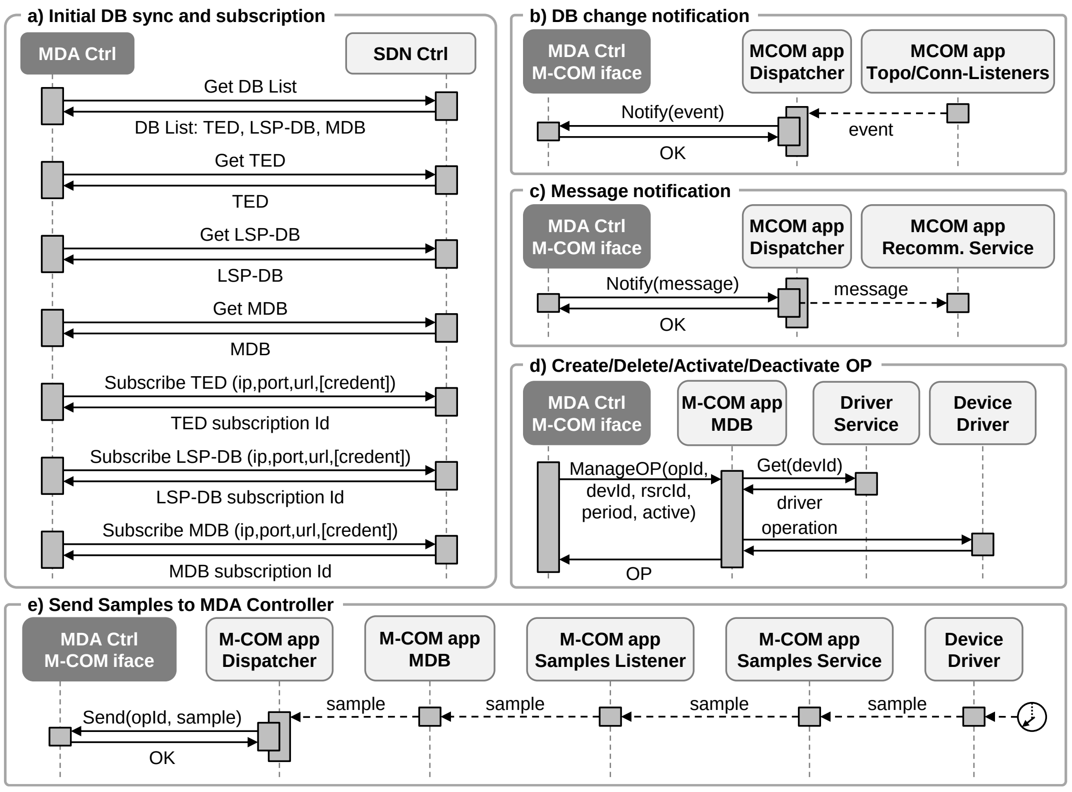

Figure 3. Operations supported by M-COM interface

At the MDA controller side, the M-COM interface consists of: *i*) a *REST API Client* that serves as a gateway to issue RPC commands to the COM module, *ii*) a *REST API Listener* that receives asynchronous messages from the COM module, such as database change events, and *iii*) a *Handler* block that integrates the logic of the M-COM interface. The latter, in turn, consists of four sub-blocks: *i*) the *Operational Databases* stores a clone of the operational databases in the COM module and is updated by database change events from the *REST API Listener*, *ii*) the *Manage Subscriptions* block deals with subscriptions handling to receive database changes from the COM, *iii*) the *Manage ObsPoints* block provides an interface to (de)configure OPs at the COM from the MDA controller, and *iv*) the *Send Recommendation* block enables the MDA controller to issue messages to feed applications in the COM module. Figure 3 illustrates the workflows for the functionalities related to the M-COM interface; these are:

- Get list of databases (*Get DB List* in a): returns the list of identifiers for those operational databases managed by a COM module.
- Get database (*Get TED* / *Get LSP-DB* / *Get MDB* in a): returns the contents of the operational database instance identified by the provided input(s).
- Subscribe to database changes (*Subscribe TED* / *Subscribe LSP-DB* / *Subscribe MDB* in a): subscribes the MDA controller to asynchronous notifications of database changes to avoid periodic polling processes. The requests should carry the endpoint (IP address, port, and URL) as well as the credentials that the M-COM application will use to issue the database change notifications. Similarly, there are unsubscribe methods to remove existing subscriptions.
- Notify database change (b): notifies about changes in a resource within a database. Operational database change events are collected using topology and connection listeners.
- Notify message (c): notifies messages from the MDA controller to the COM by means of *Recommendation Service* that distributes the message to listeners, i.e., other COM applications, subscribed to this service.
- Manage observation points (d): Creates/gets/updates/removes OPs in the COM module and triggers the appropriate operations in the related network/IT devices through COM device drivers.

- Send Samples (e): when this functionality is available at a device driver, the driver periodically collects telemetry samples from the device and issues them to the *Samples Service* from the M-COM application. Since M-COM application implements a *Samples Listener*, each sample will be received, processed by the *MDB* block and forwarded to the MDA controller through the M-COM interface. In case this functionality is not available in the driver, the device needs to be configured to forward the samples to some component in the MDA platform, e.g., an MDA agent or the MDA controller.

4 IO4 INTERFACE

The IO4 interface is designed to enable an external system, such as an OSS/BSS, to access the collected data repository at the MDA controller to retrieve raw data from network/IT devices, i.e., for monitoring, billing, or analysing different scenarios. Figure 4 illustrates the components belonging to the IO4 interface. From the MDA controller side, the interface consists of a *REST API server* that receives commands from the external system and triggers the execution of tasks in a *Requests Dispatcher*. The *Requests Dispatcher* accesses to the *MDB* to retrieve OPs and validate the requests, and to the *Samples Repository* to retrieve the samples according to the request parameters, i.e., the time frame and the OP identifier.

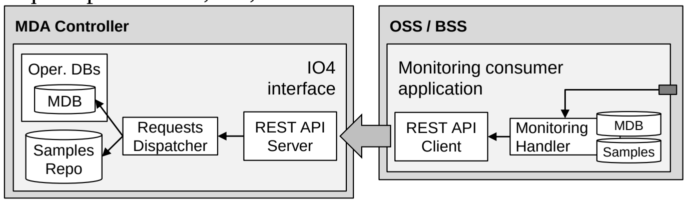

Figure 4. Components of the IO4 interface

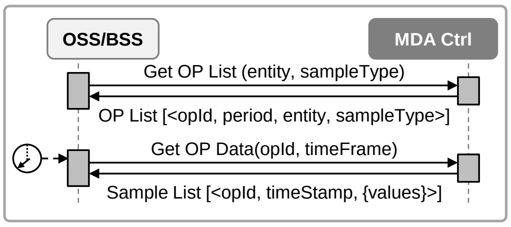
OSS/BSS
MDA Ctrl
Get OP List (entity, sampleType)
OP List [<opId, period, entity, sampleType>]
Get OP Data(opId, timeFrame)
Sample List [<opId, timeStamp, {values}>]

Figure 5. Operations of the IO4 interface

In a generalist way, an external system willing to use the IO4 interface only needs to implement a simple *REST API client* and a *Monitoring Handler* block responsible for managing the monitoring entities in the MDA controller. That *Monitoring Handler* needs to implement two databases: an *MDB* to store the attributes retrieved for each OP requested to the MDA controller, and a *Samples* database to store the samples belonging to these OPs. Note that when an OP is requested, the attributes should carry OP's sampling period; the *Monitoring Handler* at the external system should schedule sample requests with that specific periodicity to prevent flooding the MDA controller with unneeded/redundant requests.

Figure 5 illustrates the workflows defined for the functionalities related to the IO4 interface; these functionalities are briefly described below:

- Get list of OPs (*Get OP List*): returns the list of OPs that fulfill a set of optional input criteria, i.e., entity name and sample type.
- Get OP data (*Get OP Data*): returns the list of available samples for a specific OP. Optionally, a time frame can be defined for filtering purposes.

5 CONCLUSIONS

East-west interface M-COM and NBI IO4 for MDA platforms have been presented in this paper. While the M-COM interface enables interaction between a COM module, such as an SDN controller, to synchronize the operational databases, issue recommendations and configure/collect monitoring samples, the IO4 is provided to export collected monitoring samples to external systems, such as an OSS/BSS, needing to consume monitoring data for other purposes including, among others, billing and infrastructure forecasting. A conceptual explanation and requirements for them, as well as relevant workflows related to each funcionality have been described.

ACKNOWLEDGEMENTS

This work was partially supported by the EC through the METRO-HAUL project (G.A. nº 761727), from the AEI/FEDER TWINS project (TEC2017-90097-R), and from the Catalan ICREA Institution.

REFERENCES

[1] D. Rafique and L. Velasco, "Machine Learning for Optical Network Automation: Overview, Architecture and Applications," IEEE/OSA J. of Optical Communications and Networking (JOCN), vol. 10, pp. D126-D143, 2018.
[2] L. Velasco *et al.*, "A Control and Management Architecture Supporting Autonomic NFV Services," Springer Photonic Network Communications, vol. 37, pp. 24-37, 2019.
[3] Ll. Gifre *et al.*, "Autonomic Disaggregated Multilayer Networking," IEEE/OSA J. of Optical Communications and Networking (JOCN), vol. 10, pp. 482-492, 2018.
[4] L. Velasco *et al.*, "An Architecture to Support Autonomic Slice Networking [Invited]," IEEE/OSA J. of Lightwave Technology (JLT), vol. 36, pp. 135-141, 2018.