CASTOR: A Monitoring and Data Analytics Architecture to Support Autonomic Domain and Slice Networking

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

ABSTRACT
Network slices combine resource virtualization with the isolation level required by future 5G applications. In addition, the use of Monitoring and Data Analytics (MDA) helps to maintain the required network performance, while reducing total cost of ownership. In this paper, we present CASTOR, an architecture to enable autonomic domain and slice networking. MDA agents use data analytics to make local decisions close to network devices, whereas MDA controllers collate and export metered data transparently to customer controllers, all of them leveraging customizable and isolated data analytics processes. Discovered knowledge can be applied for both proactive and reactive network slice reconfiguration, triggered either by service providers or customers, thanks to the interaction with state-of-the-art software-defined networking controllers and planning tools.
Keywords: network slicing, autonomic networking, cognitive networking, monitoring and data analytics.

1. INTRODUCTION

Network slicing is one of the building blocks to support the digital transformation promised by 5G [1]. The convergence of new verticals such as Internet of Things (IoT), mobile broadband or media networks with traditional data networks is forcing operators to change architectures supporting current deployments to the Telecom cloud [2]. In fact, it is not a trivial task to accommodate the myriad of use cases and requirements demanded by such verticals, e.g., ultra-low latency response or high service availability, over a common network infrastructure. The concept of *network slicing* is closely related to network virtualization to create virtual (logical) networks decoupled from the underlying physical network. The specific feature behind network slicing is the focus on an end-to-end and service-oriented view of the network, even considering service deployment over multiple network segments from different providers.

The process of network abstraction presents the connectivity graph in a way that is independent of the underlying network domains, so it can be used to create a single virtualized network that is under the control of a customer. Currently, the IETF is working to specify the set of management and control functions to provide an abstraction of networking resources in the context of the Abstraction and Control of Traffic Engineered Networks (ACTN) framework [3]. ACTN framework extends the concept of software-defined networking (SDN) to consider network and service abstraction, as well as coordination of resources across multiple domains and layers. From an operational perspective, a network slice consists of a set of network resources and instance-specific policies and configurations that govern resources' behavior creating a complete instantiated logical network to meet certain network requirements. Since network slices might require stringent requirements, e.g., ultra-low latency, they need to be isolated from other traffic or applications in the network to guarantee the committed performance. In addition, trust in the integrity of devices and data privacy and secure communications are required.

To minimize dependency on human administrators, the concept of *autonomic* networking entails defining closing control loops aiming at providing self-management capabilities to the network [4]; this includes: *i*) monitoring resources in the network nodes, *ii*) bringing data analytics techniques to the network nodes, as well as deploying data analytics in centralized systems aiming at discovering knowledge from data (Knowledge Discovery from Data, KDD), and *iii*) using discovered knowledge for self-management purposes.

In this paper, we present CASTOR architecture to support autonomic domain and slice networking. Both network operators and customers can implement its own business intelligence and efficiently manage their own resources based on Big Data-backed data analytics techniques, isolated from other slices, through domain controllers and customer network controllers (CNCs), respectively. Consequently, total costs of ownership (TCO) will be kept minimal and revenues will increase while provisioning higher QoS services. The proposed architecture is first motivated in Section 2, where management architecture and slicing concepts are presented, and detailed in section 3. The architecture supports network slices by enforcing a clear separation of resources, where monitoring data is generated and consumed by its owner. Finally, section 4 draws the main conclusions.

2. NETWORK SLICING

Figure 1a presents a scenario based on the ACTN framework, where domains (metro and core) support network slicing. Each domain controller system includes: *i*) the provisioning and reconfiguration module based on SDN, *ii*) a data analytics module that collects data records from the nodes and runs KDD algorithms, and *iii*) a slice manager that exports virtualized network resources in the form of network slices through a northbound interface (NBI) to the CNCs. The NBI enables not only connection provisioning and network slice reconfiguration but

also monitoring the network slice, so CNCs can apply KDD algorithms for autonomic slice networking. CNCs are able to manage an end-to-end network composed of multiple network slices from multiple domain network controllers, as well as customer-owned infrastructure. They consist of an SDN controller for connection provisioning, as well as a data analytics module running KDD algorithms.

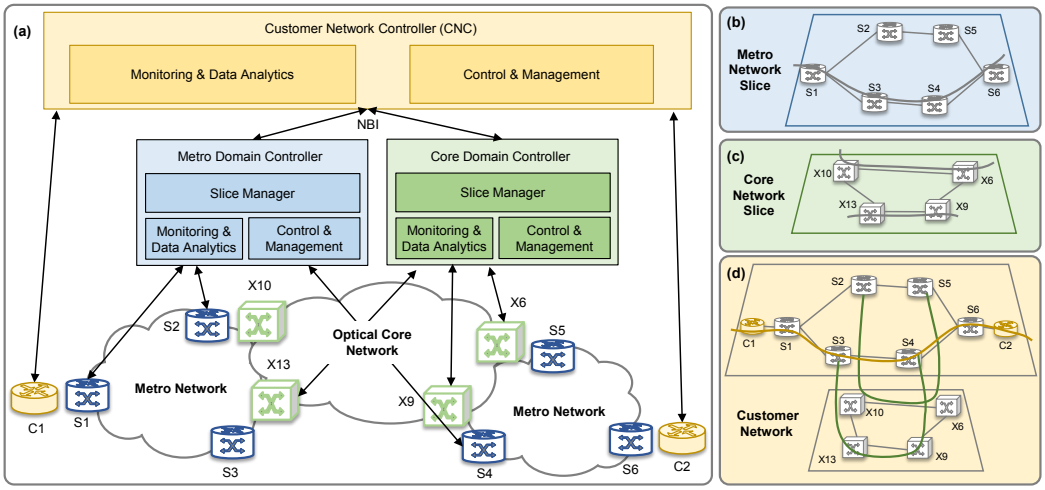

Figure 1. Management architecture and network slice concept.

In the example in Fig. 1, the metro domain controller exports a metro network slice (Fig. 1b) that includes six MPLS switches (S1..S6) connected through vlinks. Besides, the core domain controller exports a core network slice (Fig. 1c) that consists of four cross-connects (X6, X9, X10, and X13). Two lightpaths are established in the core network slice to support vlinks connecting S2-S5 and S3-S4 in the metro network slice. An MPLS connection entering through S1 and leaving through S6 is established on the metro network slice. Figure 2 shows the mapping of the core network slice on the physical core network (an example of a realistic Spanish Telefonica network is used for illustrative purposes), where the actual route of the lightpaths is represented. The exported resources allow the CNC to operate a multi-layer, multi-domain end-to-end network (Fig. 1d) that includes the two network slices and two customer nodes (C1 and C2). An end-to-end customer MPLS connection is established from C1 to C2.

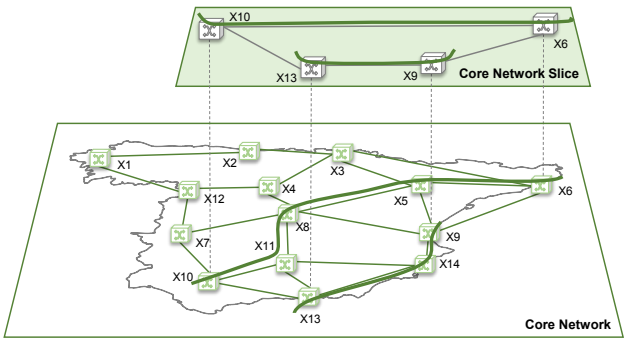

Figure 2. Core network slice physical mapping.

To monitor the physical networks and enable autonomic networking, observation points need to be configured in network nodes and metered data collated by the corresponding domain controller. In addition, to enable autonomic slice networking, observation points need to be configured in network nodes to monitor each network slice independently and metered data collated by the corresponding CNC transparently, where domain controllers play the role of IP Flow Information eXport (IPFIX) mediators [5].

3. CASTOR: ARCHITECTURE TO SUPPORT AUTONOMIC DOMAIN AND SLICE NETWORKING

Figure 3 overviews the proposed architecture, named as CASTOR, includes a number of *MDA agents* running close to the network nodes, and a big data centralized MDA controller running in the control and management plane in charge of managing the MDA agents. The number of MDA agents may vary depending on the size of the network, geographical extension, and/or any other criteria. MDA agents can be deployed as separate elements or run inside those physical nodes with enough computation capabilities. We use a different identifiers Id for each network slice and map them to different observation domains to maintain isolation between slices, and reserve observation domain Id 0 for operator's resources (named as Slice0), i.e., resources not associated with any slice, e.g., an operator's connection or a topological element like an optical link.

MDA agents (see details in Fig. 4b) are multi-node agents designed to configure monitoring and telemetry, as well as to collect monitoring data records from configured observation points in the nodes, which can be used for KDD to proactively implement local control loops to tune parameters in the network devices and to notify the MDA controller about network anomalies and degradations. MDA agents consist of two building blocks, the local configuration module and the local KDD module.

The local configuration module is in charge of receiving configuration requests from the MDA controller and interacting with the network nodes. It exposes a RESTCONF-based NBI to the MDA controller based on

a YANG data model that includes two subtrees: *i*) *applications*, for configuring the applications in the local KDD module inside the MDA agent, and *ii*) *nodes*, for configuring the underlying network devices. Finally, a number of *node agents* (one per node) are used to manage the different network nodes. Different network nodes can use different protocols for configuration, monitoring, and telemetry (e.g., NETCONF, IPFIX, and gRPC-based protocols [6] for configuration, monitoring and telemetry, respectively, or any other combination). For this very reason, node agents include *node adapters* implementing specific protocols and function mapping for the underlying network nodes.

The local KDD module contains KDD applications in charge of handling and processing data records; a different local KDD application is created for each network slice, i.e., there is a one-to-one correspondence between a KDD application and an observation domain, where both share the same Id. To isolate KDD application execution, each KDD application runs inside a container. A KDD manager is the entrance point for KDD applications; it receives IPFIX data records and delivers them to the corresponding KDD application. The KDD manager contains four main components: *i*) a container driver managing containers life-cycle; *ii*) a gRPC [6] speaker, in charge of the communication between the KDD manager and the KDD application manager; *iii*) an applications engine that routes messages (configuration, samples, and notifications) exchanged in the MDA agent; and *iv*) a network slices database containing data to properly forward messages between the KDD manager and the KDD applications.

KDD applications include: *i*) an application manager that contains a gRPC speaker for KDD manager interconnection; *ii*) a repository where data records are temporarily stored; *iii*) sample handlers to deal with data aggregation and KDD processes for knowledge discovery; and *iv*) an encryption/decryption module to provide encryption to configuration messages, samples, and notifications between the controller and the KDD application. The architecture ensures that monitoring data records are only accessed and processed by software components (sample handlers and KDD processes) specifically developed for the network slice; such software components are deployed, configured, and managed directly from the controller in charge of the slice. A public API has been defined simplify development of custom processes and sample handlers.

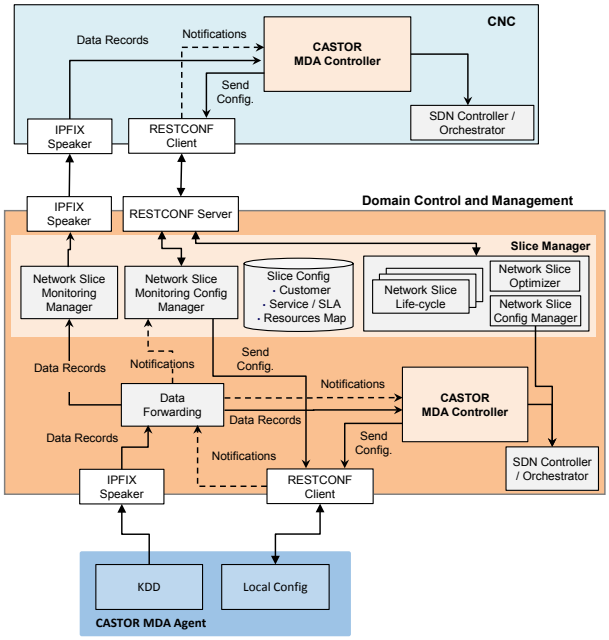

Figure 3. Proposed CASTOR architecture for autonomic domain and slice networking.

As previously stated, all messages between a KDD application and the controller of the slice can be encrypted to guarantee data privacy. To this end, both parts generate a pair of public-private keys and exchange the public key to its counterpart. For the data records and notifications, a cryptography module in the KDD application uses the controller's public key to encrypt them before being conveyed. For configuration, the module uses extended node's private key to decrypt received messages.

Regarding monitoring, the granularity of data records received from physical nodes is generally finer than that used to export data toward the domain system and therefore, data records are temporally stored and aggregated; this opens the opportunity to apply data analytics techniques directly in the network nodes. Hence, upon the reception of monitoring data records from an observation point, the KDD application manager looks at the message mapping database to find the sample handler in charge of aggregating data records of the given type, stores them in the observation point's temporal repository, and calls the KDD process in charge of processing those data records. In case the KDD process discovers a pattern in data, a notification to the controller can be sent. Periodically, data records in temporal repositories are aggregated and sent toward the controller.

The domain controller system includes the SDN controller, the domain Monitoring and Data Analytics (MDA), and the slice manager. Data records and notifications from the MDA agents are received by a data forwarding module that delivers them either to the domain MDA module in case the associated resource is locally managed (i.e., it belongs to Slice0), or to the slice manager in case the associated resource belongs to a network slice. Upon the reception of a data record or a notification from the data forwarder, the slice manager finds the CNC in charge of the network slice the associated resource belongs to and forwards the data record or notification to such CNC through the appropriate interface.

Upon the reception of network slice data records and notifications in the MDA module (its details are presented in Fig. 4a); a data manager decrypts their contents using the controller's private key to obtain plain data. Data records can be aggregated using a specific observation group handler and stored into a scalable multi-

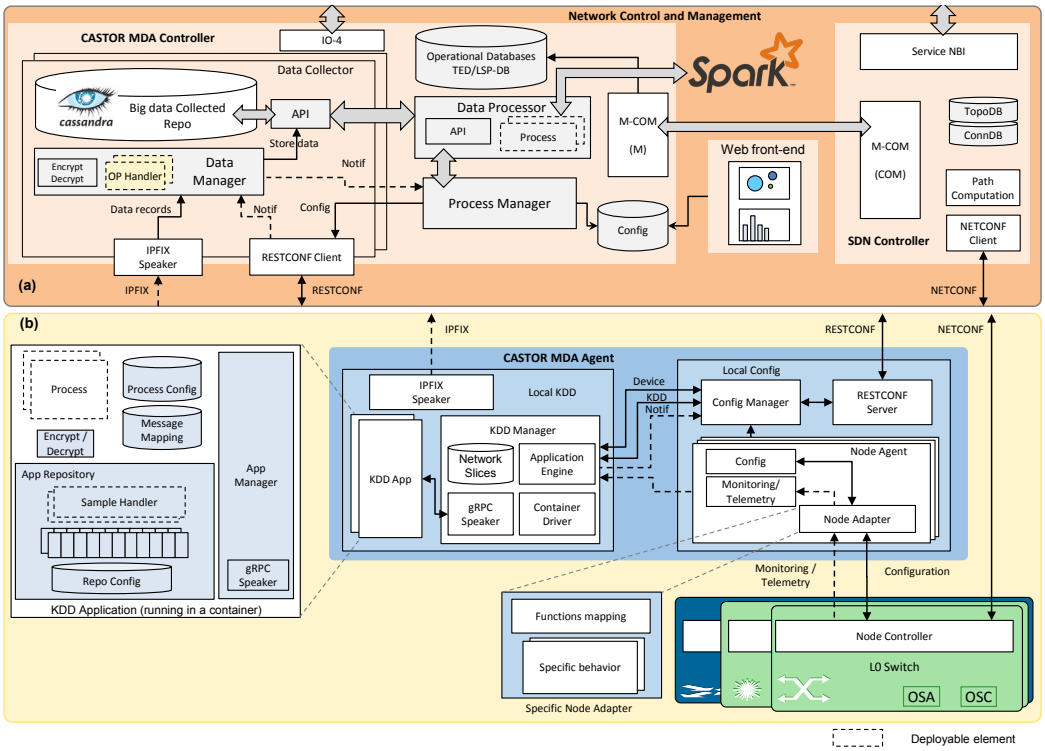

Figure 4. CASTOR architecture details: a) MDA controller and b) MDA agent.

master database (in our implementation we use Apache Cassandra). A decision maker module is notified, and the corresponding KDD process is executed; KDD processes can run locally or in a cluster using big-data processing engines, such as Apache Spark. Additionally, the controller manages the configuration of the KDD application deployed in the network nodes; whenever configuration parameters need to be tuned, a message encrypted using the KDD application's public key can be sent through the RESTCONF interface. Similarly as for the MDA agent, a public API has been defined to simplify processes and observation group handlers development. In the case that a network slice reconfiguration is needed, the SDN controller can be triggered.

4. CONCLUSIONS

Network slicing combines virtualization and isolation to support 5G applications with ultra-low latency and high availability. In this paper, CASTOR architecture to enable autonomic slice networking has been presented. The architecture consists of a number of MDA agents connected to their physical counterparts. Each MDA agent consists of two main building blocks: *i*) a local configuration block in charge of configuring monitoring and telemetry features of physical network nodes connected to the MDA agent, as well as collecting monitoring and telemetry data; and *ii*) a local KDD block responsible for the data processing and analysis of the collected monitoring and telemetry data, thus enabling making local decisions. The domain MDA collects monitoring data and distributes them for either domain analysis or customer analysis, depending on whether the network resource is locally managed, or it belongs to a network slice managed by a customer controller. In both cases, big data-based data analytics techniques can be applied to discover knowledge from data and to implement network and slice-wide closed control loops.

ACKNOWLEDGEMENTS

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).

REFERENCES

[1] J. Ordonez-Lucena *et al*., "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges*," IEEE Communications Magazine*, vol. 55, pp. 80-87, 2017.
[2] L. Velasco *et al*., "A service-oriented hybrid access network and cloud architecture," *IEEE Communications Magazine*, vol. 53, pp. 159-165, 2015.
[3] D. Ceccarelli and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks," IETF draft, work-in-progress, 2018.
[4] M. Behringer *et al*., "Autonomic Networking: Definitions and Design Goals," IETF RFC 7575, 2015.
[5] B. Claise, A. Kobayashi, and B. Trammell, "Operation of the IP Flow Information Export (IPFIX) Protocol on IPFIX Mediators," IETF RFC 7119, 2014.
[6] gRPC Remote Procedure Call: http://www.grpc.io/