First Experimental Demonstration of Autonomic Slice Networking

Luis Velasco[1*], Lluís Gifre[2], Francesco Paolucci[3], and Filippo Cugini[4]

[1] *Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain*
[2] *Universidad Autónoma de Madrid (UAM), Madrid, Spain,* [3] *Scuola Superiore Sant'Anna, Pisa, Italy,* [4] *CNIT, Pisa, Italy*
e-mail:lvelasco@ac.upc.edu

Abstract: An architecture to enable autonomic slice networking is presented. Extended nodes make local decisions, whereas domain systems collate and export metered data transparently to customer controllers. Discovered knowledge is experimentally used for network slice reconfiguration.

OCIS codes: (060.4250) Networks; (060.4261) Networks, protection and restoration

1. Introduction

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 application in the network to guarantee the committed performance [1]. 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 closing control loops aiming at providing self-management capabilities[2]; we call this as the *observe-analyze-act* (OAA) loop [3] since it includes: *i*) monitoring resources in the network nodes, *ii*) using data analytics techniques both, locally in the network nodes, e.g., to detect anomalies and degradations [4], and in a centralized system aiming at discovering knowledge from data (KDD), and *iii*) using discovered knowledge for self-management purposes.

This paper presents and experimentally validates for the first time an architecture to support autonomic slice networking, so services can implement their own business intelligence and, through the Customer Network Controller (CNC), can efficiently manage their resources based on the data analysis outcome. This will allow keeping Total Cost of Ownership (TCO) minimal while provisioning higher QoS services.

2. Network Slicing

Fig. 1a presents a scenario based on the ACTN framework [5], where two domains for metro and core support network slicing. Each domain control/management system includes: *i*) the provisioning and reconfiguration module, e.g., based on ABNO, *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 a module for connection provisioning and network slice re-configuration, as well as a data analytics module running KDD algorithms.

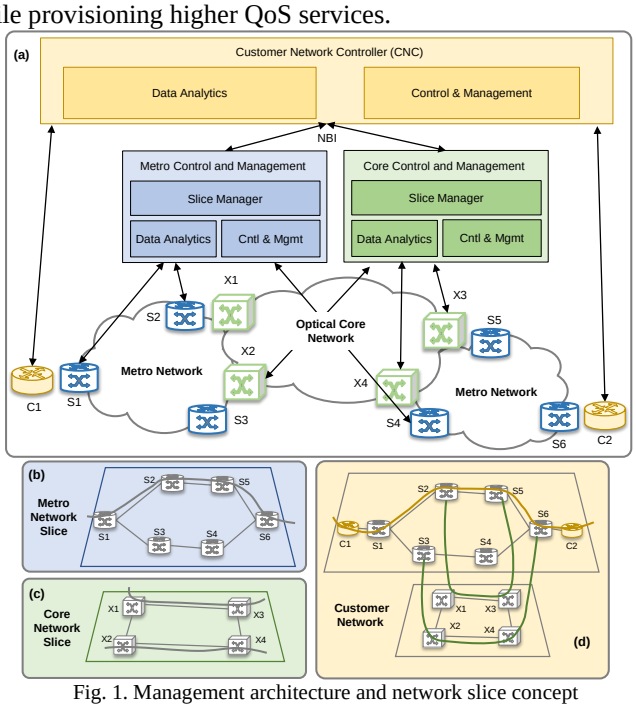

Fig. 1. Management architecture and network slice concept

In the example in Fig. 1, domain controllers exporting network slices allow the CNC to operate a multi-layer, multi-domain network that includes both sliced and customer-owned physical resources. The metro network slice (Fig. 1b) includes six MPLS switches (S1..6) connected through virtual links (vlink), where vlinks S2-S5 and S3-S4 in particular, are supported by lightpaths through the core network slice. The core network slice (Fig. 1c) consists of four cross-connects (X1..4), where two lightpaths are established. An MPLS connection entering through S1 and leaving through S6 is established on the metro network slice. The CNC operates an end-to-end network (Fig. 1d) that includes the network slices and two customer nodes (C1 and C2). An end-to-end

The research leading to these results has received funding from the Spanish MINECO SYNERGY project (TEC2014-59995-R), from the Catalan Institution for Research and Advanced Studies (ICREA), and from the EU Commission for the H2020-ICT-2014 project 5GEx (G.A. 671636).

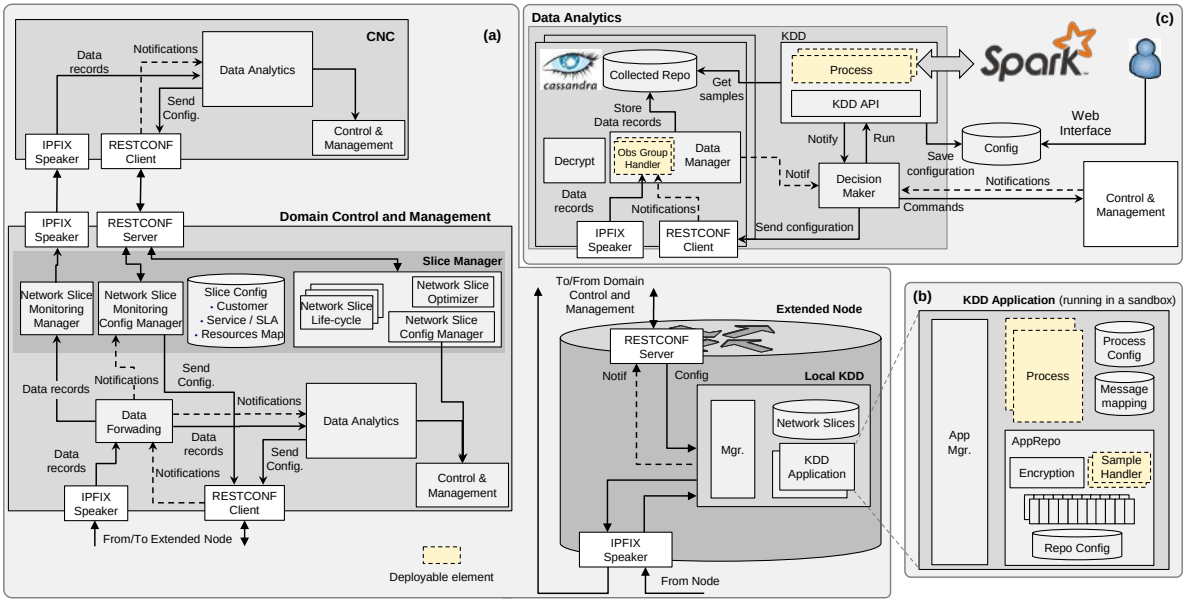

Fig. 2. Detailed architecture proposed for Autonomic Slice Networking.

customer MPLS connection is established from C1 to C2. To enable autonomic slice networking, *observation points* need to be configured in network nodes for each network slice independently and metered data collated by the corresponding CNC transparently, where domain systems play the role of *IPFIX mediators* [6].

3. Architecture to support Autonomic Slice Networking

Fig. 2a overviews the proposed architecture. Extended node modules receive IPFIX messages from physical nodes containing data records with data from observation points belonging to an *observation domain Id*; we use a different Id for each network slice. The extended node includes a different local KDD application for each network slice in charge of handling and processing data records. A manager is the entrance point for the KDD applications; it receives data records and delivers them to the corresponding KDD application. Extended nodes can be deployed as separate elements or run inside physical nodes with computation capabilities. KDD applications (Fig. 2b) include: *i*) a repository where data records are temporarily stored; *ii*) a number of software components (KDD processes and data handlers to deal with data aggregation); and *iii*) an application manager that serves as an interface between the KDD application and the rest of elements. The architecture ensures that monitoring data records are only accessed and processed by software components specifically developed for the network slice; such software components are deployed directly from the CNC managing the network slice. To isolate KDD application execution, each KDD application runs in a self-contained execution environment. In addition, an encryption module uses the CNC' public key to encrypt data records being conveyed to the CNC.

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 process 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 the data, a notification to the CNC controller can be sent. Periodically, data records in each temporal repository are aggregated, encrypted, and sent toward the CNC.

The domain control/management system includes the domain analytics and the slice manager. Data records and notifications received from the network nodes are received by a *data forwarder* module that delivers them either to the domain analytics module in case the associated resource is locally managed, 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 managing the network slice the associated resource belongs to and forwards the data record or notification to such CNC through the appropriate interface.

Data records and notifications arriving at the CNC are sent to the analytics module (Fig. 2c); a *data manager* decrypts their contents using the CNC's private key to obtain plain data. Data records can be aggregated using a specific sample handler and stored into a scalable multi-master database. A decision maker module is notified, and the corresponding CNC KDD process is executed; KDD processes can run locally or in a cluster using big-data processing engines. In the case that a network slice reconfiguration is needed, the CNC network controller can be triggered. Note also that the CNC 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 with the new values.

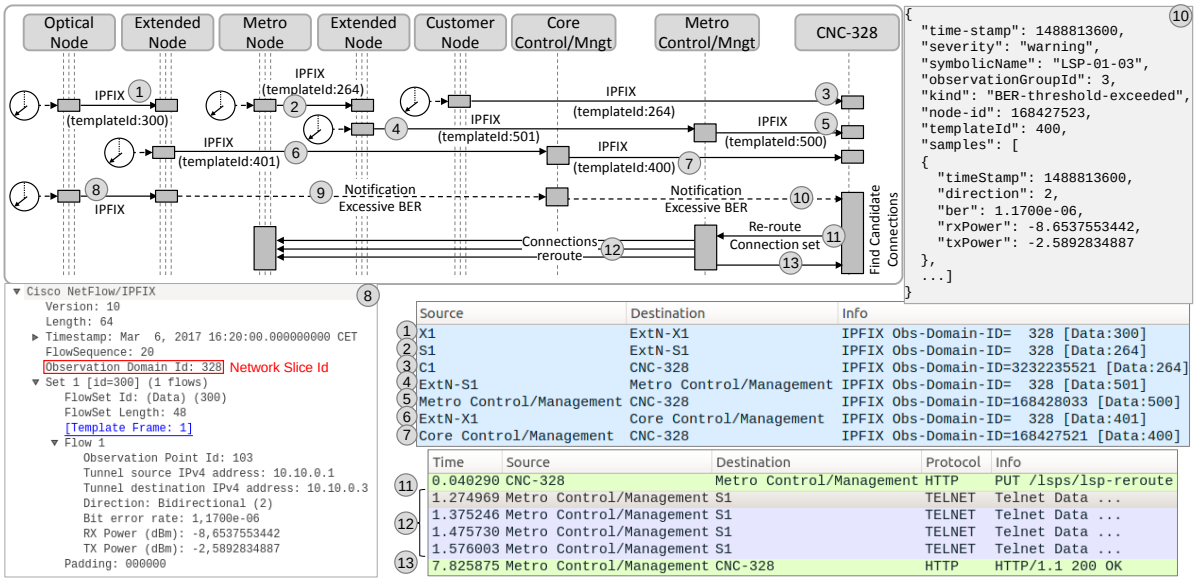

Fig. 3. Workflow and captures.

4. Experimental assessment

For the experimental validation, we implemented a network slice reconfiguration use case, where the CNC decides to reroute those MPLS connections using vlink X1-X3 when BER degradation on the underlying lightpath is detected by the KDD process in the related extended node. Experiments have been carried out on a distributed field trial set-up connecting premises in UPC (Barcelona, Spain), and Scuola Superiore Sant'Anna (Pisa, Italy) through IPSec tunnels. UPC's extended node, network slice manager and data analytics modules were implemented in Python and run in a computer cluster under Linux. UPC's CNC control and management module was developed in Python and interoperates with a planning tool developed in C++ for Linux. The data plane is realized at Sant'Anna premises, and it consists of a network including commercial Juniper routers equipped with GbE interfaces and Ericsson SPO-1400 ROADMs encompassing 100Gb/s transceiver cards. Nodes are controlled by C++-based SDN controllers handling configuration by means of dedicated southbound interfaces [7]. The NBI exploits a REST API server able to receive vlink and connections' setup and modify requests from the CNC, along with TED and LSP-DB synchronization procedures.

The scenario depicted in Fig. 1 was reproduced; Fig. 3 presents the OAA loop implemented. Different IPFIX templates were used to encode monitoring data, depending on the type of observation point, and the IPFIX session, and its data can be optionally encrypted. We defined the template Id 300 to encode optical transponder monitoring data, including BER and optical power (message 1) and used OpenVSwitch's template Id 264 for L2 traffic (message 2); the field *Observation Domain Id* is used to identify the network slice. We configured L2 and L0 nodes to send monitoring data records every 60 s. Extended nodes use a different template to aggregate data and were configured to send data records every 15 min.; those templates include the originator *nodeId* as *original Observation Domain Id* (template Ids 401 and 501) (messages 4 and 6). Finally, the domain control/management system uses templates IDs 400 and 500 with the CNCs, where the field *Observation Domain Id* contains the originator *nodeId* (messages 5 and 7) and slicing information was removed. Note that customer's nodes send IPFIX monitoring data directly to the CNC (message 3).

Upon a BER degradation (we configured a BER threshold equal to 1e-6 for lightpath X1-X3) is detected in an extended node (after reception of message 8), a notification that includes the last measured values on the observation point is sent to the network slice's CNC, via the core control and management system (messages 9 and 10). The CNC decides to re-route customer connections using the degraded vlink, so it collects all the affected connections and requests their rerouting to the metro domain control/management system, excluding the vlink in degraded BER condition from the path computation (messages 11-13).

5. Conclusions

An architecture to enable autonomic slice networking has been presented and experimentally demonstrated using a use case of preventive network slice MPLS connection rerouting in the event of excessive BER detected in a lightpath belonging to the network slice that supports a vlink.

References

[1] NGMN 5G P1 "Requirements & Architecture Work Stream End-to-End Architecture: Description of Network Slicing Concept," 2016.

[2] M. Behringer et al., IETF RFC 7575, 2015.

[3] Ll. Gifre et al., "Experimental Assessment of Node and Control Architectures to Support the Observe-Analyze-Act Loop," OFC 2017

[4] A. P. Vela et al., "Early Pre-FEC BER Degradation Detection to Meet Committed QoS," OFC, 2017.

[5] D. Ceccarelli and Y. Lee, draft-ceccarelli-teas-actn-framework, IETF work-in-progress, 2017.

[6] B. Claise et al., IETF RFC 7119, 2014.

[7] F. Paolucci et al., "Service Chaining in Multi-Layer Networks using Segment Routing and Extended BGP FlowSpec", OFC 2017.