An Architecture to Support Autonomic Slice Networking [Invited]

L. Velasco, Ll. Gifre, J.-L. Izquierdo-Zaragoza, F. Paolucci, A. P. Vela, A. Sgambelluri, M. Ruiz and F. Cugini

***Abstract*—Network slices combine resource virtualization with the isolation level required by future 5G applications. In addition, the use of monitoring and data analytics help to maintain the required network performance, while reducing total cost of ownership. In this paper, an architecture to enable autonomic slice networking is presented. Extended nodes make local decisions close to network devices, whereas centralized domain systems 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. The architecture is experimentally demonstrated by means of a complex use case for a multi-domain multilayer MPLS-over-optical network. In particular, the use case consists of the following Observe-Analyze-Act loops: *i*) proactive network slice rerouting after BER degradation detection in a lightpath supporting a virtual link (vlink); *ii*) reactive core network restoration after optical link failure; and *iii*) reactive network slice rerouting after the degraded lightpath is restored. The proposed architecture is experimentally validated on a distributed testbed connecting premises in UPC (Spain) and CNIT (Italy).**

***Index Terms*—Network slicing, Autonomic networking, Cognitive networking, Monitoring and data analytics.**

I. Introduction

Network slicing is one of the building blocks to support the digital transformation promised by 5G [2]. 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 [3]. In fact, it is not a trivial task to accommodate the myriad of use cases and requirements demanded by such vertical, 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 [4] 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 [5].

Specifically, 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 [6]. ACTN framework extends the concept of software-defined networking (SDN) to consider network and service abstraction and 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 [5]. 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 [7]; we call this as the *observe-analyze-act* (OAA) loop [8] since it includes: *i*) monitoring resources in the network nodes, *ii*) bringing data analytics techniques to the network nodes, e.g., to detect traffic anomalies [9] and degradations [10], 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, e.g., in-operation network planning [11] such as virtual network topology self-adaptation to traffic changes [12] and optical network re-optimization [13].

In this paper, we extend our work in [1] to present and experimentally validate an architecture to support autonomic slice networking. Both network operators and customers can implement its own business intelligence and efficiently manage their own resources based on data analytics techniques, isolated from other slices, through domain controllers and customer network controllers (CNCs),

Manuscript received July 1, 2017.
This work was presented in part at OFC 2017 [1].
Luis Velasco (lvelasco@ac.upc.edu), Jose Luis Izquierdo-Zaragoza, Alba P. Vela and Marc Ruiz are with the optical communications group (GCO) at Universitat Politecnica de Catalunya (UPC), Barcelona, Spain. Lluis Gifre is with Universidad Autonoma de Madrid (UAM), Madrid, Spain. Francesco Paolucci and Andrea Sgambelluri are with Scuola Superiore Sant'Anna, Pisa, Italy. Filippo Cugini is with CNIT, Pisa, Italy.

respectively. Consequently, total costs of ownership (TCO) will be kept minimal and revenues will increase while provisioning higher QoS services. Particularly, the contribution of this paper is two-fold:

1) The proposed architecture is first motivated in Section II, where management architecture and slicing concepts are presented. Section III is devoted to detail the data monitoring and data analytics architecture, as well as the different modules in the architecture, including: *i*) a set of extended nodes that receive monitoring data records from network nodes and apply data analytics algorithms to make local decisions; *ii*) a hierarchy of controllers making slide-wide decisions driven by centralized data analytics engines. The architecture supports network slices by enforcing a clear separation of resources, where monitoring data is generated and consumed by its owner.
2) To demonstrate the proposed architecture, a complex use case entailing three OAA loops is defined in Section IV, as well as their workflows on the proposed architecture. The defined OAA loops are: *i*) proactive network slice rerouting after BER degradation detection in a lightpath supporting a virtual link (vlink); *ii*) reactive core network restoration after optical link failure; and *iii*) reactive network slice rerouting after normal BER conditions in a previously degraded vlink. The architecture and the use case are assessed in Section V on a distributed testbed.

II. NETWORK SLICING

Fig. 1a presents a scenario based on the ACTN framework, where two domains for metro and core support network slicing. Each domain control/management 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 enriched with a planning tool for network re-optimization, as well as a data analytics module running KDD algorithms.

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, where vlinks S2-S5 and S3-S4 in particular, are supported by lightpaths through the core network slice. An MPLS connection entering through S1 and leaving through S6 is established on the metro network slice. The core network slice (Fig. 1c) consists of four cross-connects (X6, X9, X10, and X13), where two lightpaths are established to support vlinks S2-S5 and S3-S4. Fig. 2 shows the mapping of the core network slice on the physical

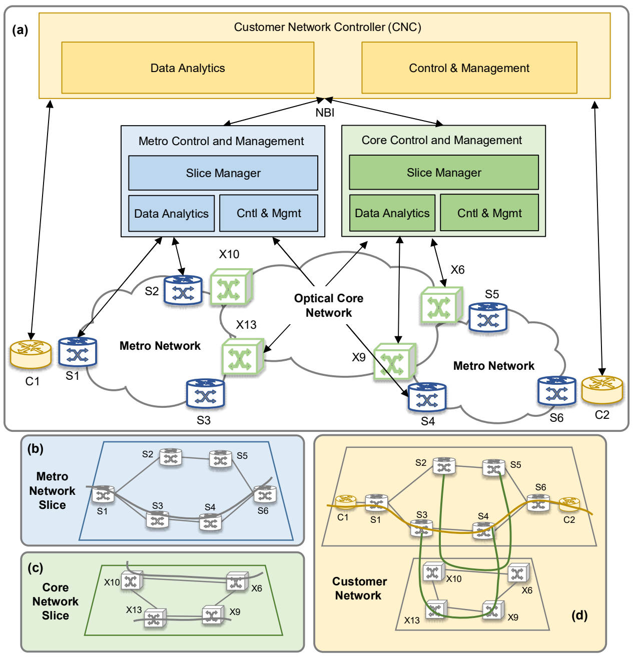


Fig. 1. Management architecture and network slice concept.

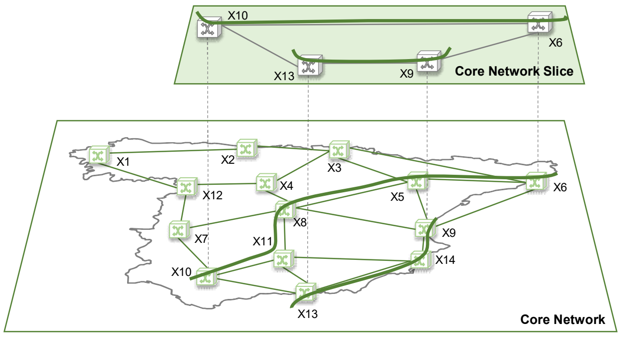


Fig. 2. Core network slice physical mapping.

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.

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 [14].

In the next section, we present an architecture to support autonomic networking at both, the domain and the slice levels.

III. ARCHITECTURE TO SUPPORT AUTONOMIC DOMAIN AND SLICE NETWORKING

Fig. 3 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 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. Extended nodes can be deployed as separate elements or run inside those physical nodes with enough computation capabilities.

The extended node (see details in Fig. 4b) includes a local KDD module that 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 [15] 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 extended node; 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.

As previously stated, all messages between a KDD application and the controller in charge 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 cryptography 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 the data, a notification to the controller can be sent. Periodically, data records in temporal repositories are aggregated and sent toward the controller.

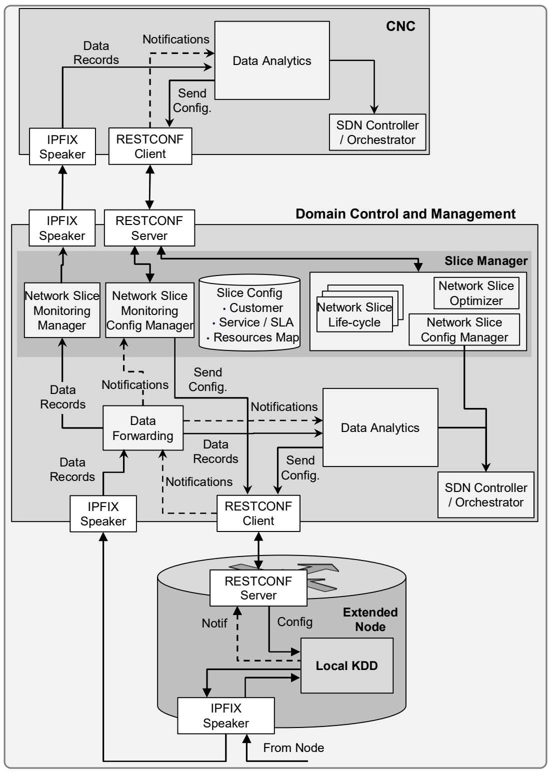


Fig. 3. Proposed architecture for Autonomic Domain and Slice Networking.

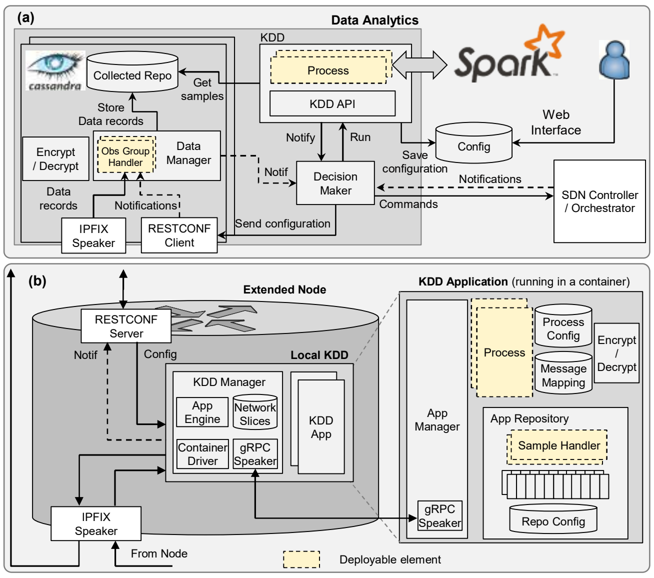


Fig. 4. Architecture details. a) Data Analytics module and b) Extended Node.

The domain control/management system includes the SDN controller, the domain data analytics, and the slice manager. Data records and notifications from the extended nodes are received by a *data forwarding* module that delivers them either to the domain analytics 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 analytics module; the details of the data analytics module 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-master database (in our implementation we use Apache Cassandra [16]). 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 [17]. 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 extended node, 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. An illustrative use case of three OAA loops entailing customer and operator network reconfiguration triggered by the changes in the status of the optical layer is presented in the next section.

IV. Demonstrative Use Case

In this section, we propose a complex use case to demonstrate how the proposed architecture works. Let us assume the scenario depicted in Fig. 1, where a customer controller with Id 328 is in charge of a network that includes two network slices, one in the core and another in the metro networks. Let us also assume that a latency-sensitive service is being supported on top of customer MPLS connection C1-C2 and thus, the route of such connection has been selected to minimize the transmission delay; specifically, the route includes vlink S3-S4.

The use case encompasses three different OAA loops that are represented as different workflows (WF) in Fig. 5: WF1) proactive network slice rerouting after BER degradation detection; WF2) reactive core network restoration after optical link failure; and WF3) proactive network slice rerouting after normal BER notification (because of lightpath restoration).

Starting with WF1 in Fig. 5, imagine that because of some problem affecting optical link X13-X14, a degradation condition arises and an increasing BER is measured in the reception of every lightpath using such link. In particular, observation points configured in the transponders of lightpath S3-S4 measure the BER (message 1 in WF1) and a KDD process in KDD application 328, running in the extended nodes, detects BER degradation [10] and notifies CNC-328 about such excessive BER (message 3), via the core domain controller. Upon receiving the notification, the CNC proactively decides to reroute MPLS connection C1-C2 through vlink S2-S5, which increases transmission delay.

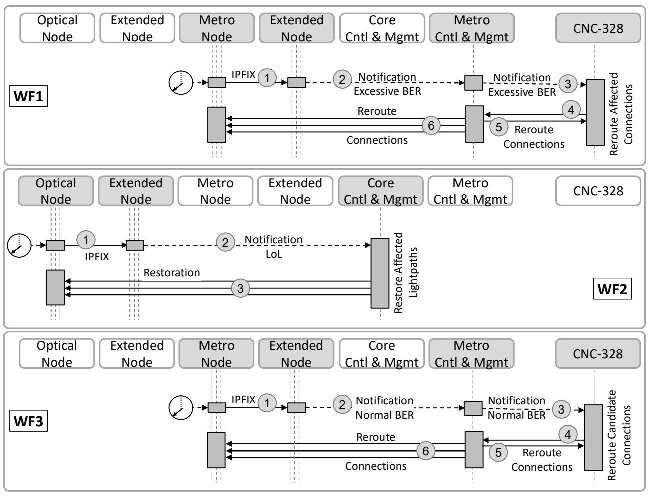


Fig. 5. Workflows demonstrated in this paper.

In case of optical link X13-X14 continues degrading, it possibly will be totally disconnected; in WF2, this event is immediately sensed by photodetectors in the end nodes of the link. In such case, the node sends an IPFIX message with the power measure (message 1 in WF2) to its extended node, where the KDD application in charge of Slice0 resources generates a Loss of Light (LoL) notification to the core domain controller (message 2). Upon its reception, the domain controller triggers a reactive restoration procedure (message 3), where new routing and spectrum allocation is found for the lightpaths using link X13-X14 [18]. Note that customer network slices are not aware neither of the link failure nor the restoration process.

When the restoration process ends, lightpath S3-S4 BER gets back to normal values (message 1 in WF3), which is detected by the KDD application in the extended nodes that issues a normal BER notification to the CNC. In this case, the route for customer MPLS connection C1-C2 that includes vlink S3-S4 reduces transmission delay, so the CNC decides to reroute such connection (message 4).

The next section presents the experimental results of the proposed architecture for this complex use case.

V. Experimental Assessment

Experiments have been carried out on a distributed field trial set-up connecting premises in UPC (Barcelona, Spain), and CNIT (Pisa, Italy) through IPSec tunnels.

The data plane was realized through two metro OpenFlow domains emulated in Mininet [19] at UPC's premises, connected through a core network including commercial Juniper routers and Ericsson SPO-1400 ROADMs deployed in CNIT premises. In the control and management plane 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. Both metro and CNC controller are connected to an instance of ONOS [20] for the role of SDN controller. Besides, SDN controllers are connected to an instance of Net2Plan [21] for network re-optimization. To this end, custom extensions were developed for both frameworks extending their base services and north/south interfaces to support new inter-module interactions like notifications, path computation, and topology exportation. In brief, the developed extensions are: *i*) decision maker to ONOS east-west interface to trigger network reconfiguration (see Fig. 4a); *ii*) northbound and southbound interfaces for hierarchical ONOS communication; and *iii*) ONOS to Net2Plan east-west interface for path computation and re-optimization. Regarding the core controller, nodes are controlled by a C++-based SDN controller handling configuration by means of dedicated southbound NETCONF interfaces [22], [23].

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. Fig. 6 presents monitoring data collection, where observation points have been configured for: *i*) L0 averaged optical power for optical link X13-X14 that is encoded using IPFIX template Id 300 (message 1 in Fig. 6); *ii*) L0 optical transponder monitoring data including optical power and pre-BER for vlink S3-S4 that is encoded using IPFIX template Id 310 (message 3); and *iii*) L2 traffic data including packets and bytes since the last report for link S1-S2 that is encoded using IPFIX Open vSwitch's template Id 264 (message 6).

Nodes use the field *Observation Domain Id* to identify the network slice. We configured L2 and L0 nodes to send monitoring data records every 60s. Extended nodes use a different template Id (Ids 3x1) to aggregate data and were configured to send data records every 15 min.; those templates include the originator *nodeId* as *original Observation Domain Id* (messages 2, 4, and 7). Finally, the domain control/management system uses templates Ids 3x2 with the CNCs, where the field *Observation Domain Id* contains the originator *nodeId* and slicing information was removed (messages 5 and 8). Note that customer's nodes send IPFIX monitoring data directly to the CNC using IPFIX Open vSwitch's template Id 264 (message 9). Fig. 7 presents a capture with the described IPFIX messages and details of data records with different IPFIX templates issued by network nodes. Note that messages are numbered as in Fig. 6.

Let us now concentrate in experimentally demonstrating the workflows proposed for the use case defined in Section IV. Fig. 8 presents the messages exchanged between the involved entities in WF1, where messages are numbered as in WF1 in Fig. 5. Upon reception of an IPFIX message with a pre-BER value in vlink S3-S4 above a predefined threshold (we configured a pre-BER threshold equal to 1e-6) (message 1 in Fig. 8), the extended node issues a "threshold exceeded" notification to the CNC via the metro controller (messages 2-3). Fields tagged as "application-id" and "severity" include local KDD application identifier (328) and severity of the notification ("major"), whilst field "data" includes symbolicName and observationGroupId of the device triggering the event, kind of event, and the sample that originated the event.

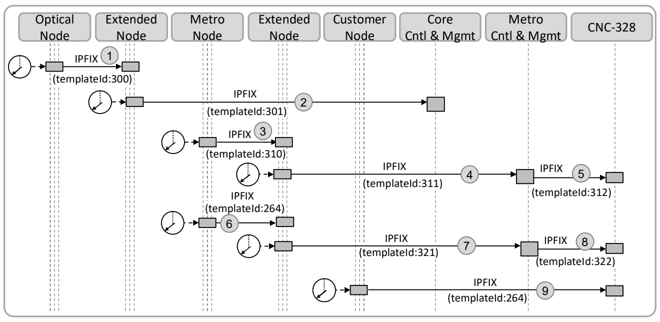
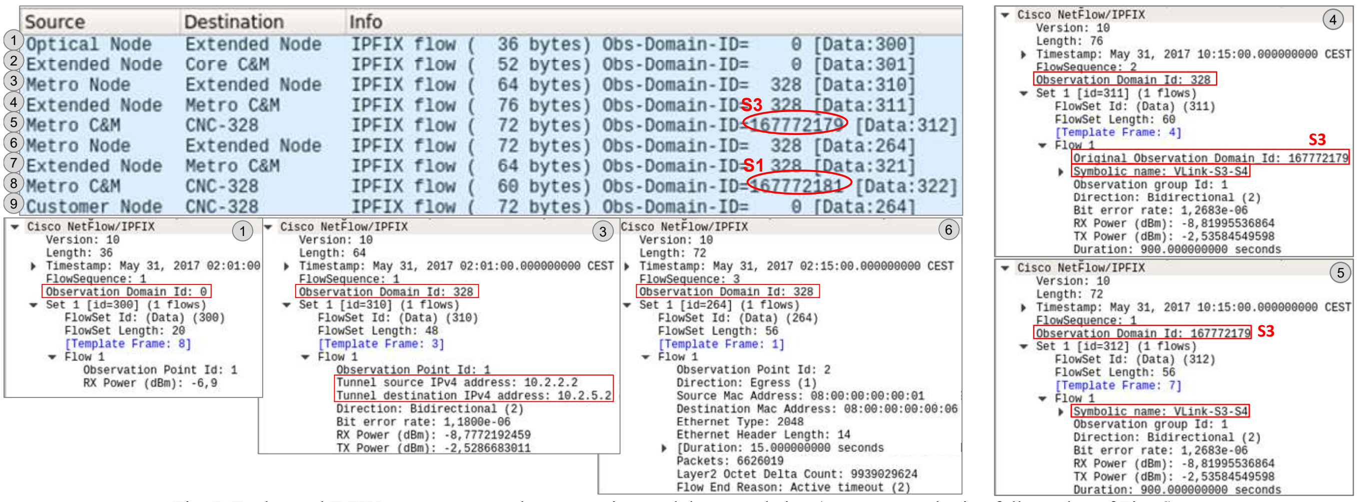


Fig. 6. Monitoring messages.


```
  Source          Destination      Info
1 Optical Node    Extended Node    IPFIX flow (  36 bytes) Obs-Domain-ID=    0 [Data:300]
2 Extended Node   Core C&M         IPFIX flow (  52 bytes) Obs-Domain-ID=    0 [Data:301]
3 Metro Node      Extended Node    IPFIX flow (  64 bytes) Obs-Domain-ID=  328 [Data:310]
4 Extended Node   Metro C&M        IPFIX flow (  76 bytes) Obs-Domain-ID=S3 328 [Data:311]
5 Metro C&M       CNC-328          IPFIX flow (  72 bytes) Obs-Domain-ID=167772179 [Data:312]
6 Metro Node      Extended Node    IPFIX flow (  72 bytes) Obs-Domain-ID=  328 [Data:264]
7 Extended Node   Metro C&M        IPFIX flow (  64 bytes) Obs-Domain-ID=S1 328 [Data:321]
8 Metro C&M       CNC-328          IPFIX flow (  60 bytes) Obs-Domain-ID=167772181 [Data:322]
9 Customer Node   CNC-328          IPFIX flow (  72 bytes) Obs-Domain-ID=    0 [Data:264]
```

```
▾ Cisco NetFlow/IPFIX                                  1
    Version: 10
    Length: 36
  ▸ Timestamp: May 31, 2017 02:01:00
    FlowSequence: 1
    Observation Domain Id: 0
  ▾ Set 1 [id=300] (1 flows)
      FlowSet Id: (Data) (300)
      FlowSet Length: 20
      [Template Frame: 8]
    ▾ Flow 1
        Observation Point Id: 1
        RX Power (dBm): -6,9
```

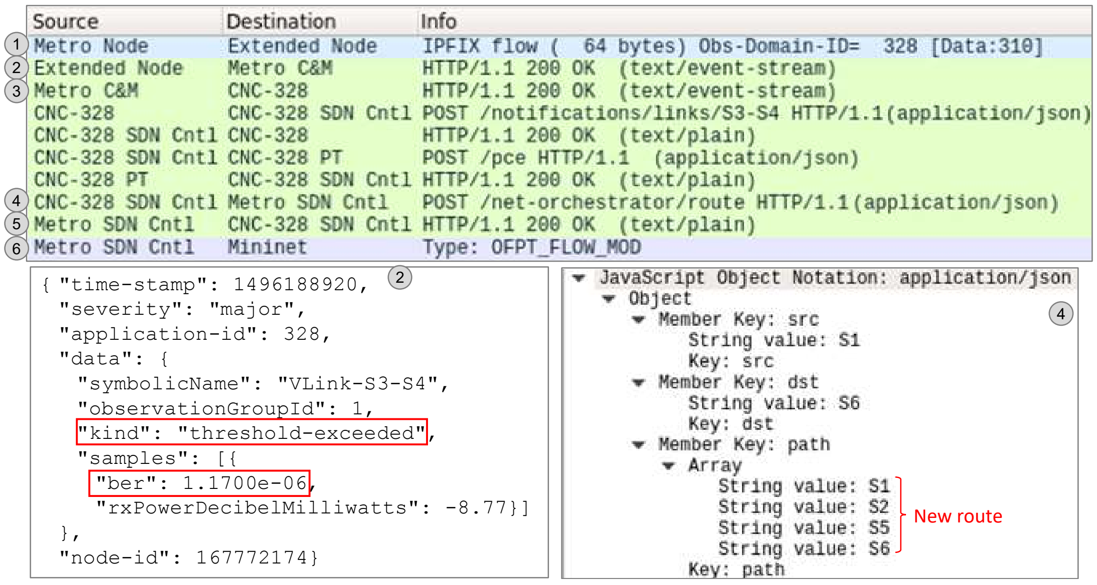
```
▾ Cisco NetFlow/IPFIX                                  3
    Version: 10
    Length: 64
  ▸ Timestamp: May 31, 2017 02:01:00.000000000 CEST
    FlowSequence: 1
    Observation Domain Id: 328
  ▾ Set 1 [id=310] (1 flows)
      FlowSet Id: (Data) (310)
      FlowSet Length: 48
      [Template Frame: 3]
    ▾ Flow 1
        Observation Point Id: 1
        Tunnel source IPv4 address: 10.2.2.2
        Tunnel destination IPv4 address: 10.2.5.2
        Direction: Bidirectional (2)
        Bit error rate: 1,1800e-06
        RX Power (dBm): -8,7772192459
        TX Power (dBm): -2,5286683011
```

```
Cisco NetFlow/IPFIX                                    6
    Version: 10
    Length: 72
  ▸ Timestamp: May 31, 2017 02:15:00.000000000 CEST
    FlowSequence: 3
    Observation Domain Id: 328
  ▾ Set 1 [id=264] (1 flows)
      FlowSet Id: (Data) (264)
      FlowSet Length: 56
      [Template Frame: 1]
    ▾ Flow 1
        Observation Point Id: 2
        Direction: Egress (1)
        Source Mac Address: 08:00:00:00:00:01
        Destination Mac Address: 08:00:00:00:00:06
        Ethernet Type: 2048
        Ethernet Header Length: 14
      ▸ [Duration: 15.000000000 seconds
        Packets: 6626019
        Layer2 Octet Delta Count: 9939029624
        Flow End Reason: Active timeout (2)
```

```
▾ Cisco NetFlow/IPFIX                                  4
    Version: 10
    Length: 76
  ▸ Timestamp: May 31, 2017 10:15:00.000000000 CEST
    FlowSequence: 2
    Observation Domain Id: 328
  ▾ Set 1 [id=311] (1 flows)
      FlowSet Id: (Data) (311)
      FlowSet Length: 60
      [Template Frame: 4]                              S3
    ▾ Flow 1
        Original Observation Domain Id: 167772179
      ▸ Symbolic name: VLink-S3-S4
        Observation group Id: 1
        Direction: Bidirectional (2)
        Bit error rate: 1,2683e-06
        RX Power (dBm): -8,81995536864
        TX Power (dBm): -2,53584549598
        Duration: 900.000000000 seconds
```

```
▾ Cisco NetFlow/IPFIX                                  5
    Version: 10
    Length: 72
  ▸ Timestamp: May 31, 2017 10:15:00.000000000 CEST
    FlowSequence: 1
    Observation Domain Id: 167772179 S3
  ▾ Set 1 [id=312] (1 flows)
      FlowSet Id: (Data) (312)
      FlowSet Length: 56
      [Template Frame: 7]
    ▾ Flow 1
      ▸ Symbolic name: VLink-S3-S4
        Observation group Id: 1
        Direction: Bidirectional (2)
        Bit error rate: 1,2683e-06
        RX Power (dBm): -8,81995536864
        TX Power (dBm): -2,53584549598
        Duration: 900.000000000 seconds
```

Fig. 7. Exchanged IPFIX messages, sample aggregation, and data translation (message numbering follows that of Fig. 6).

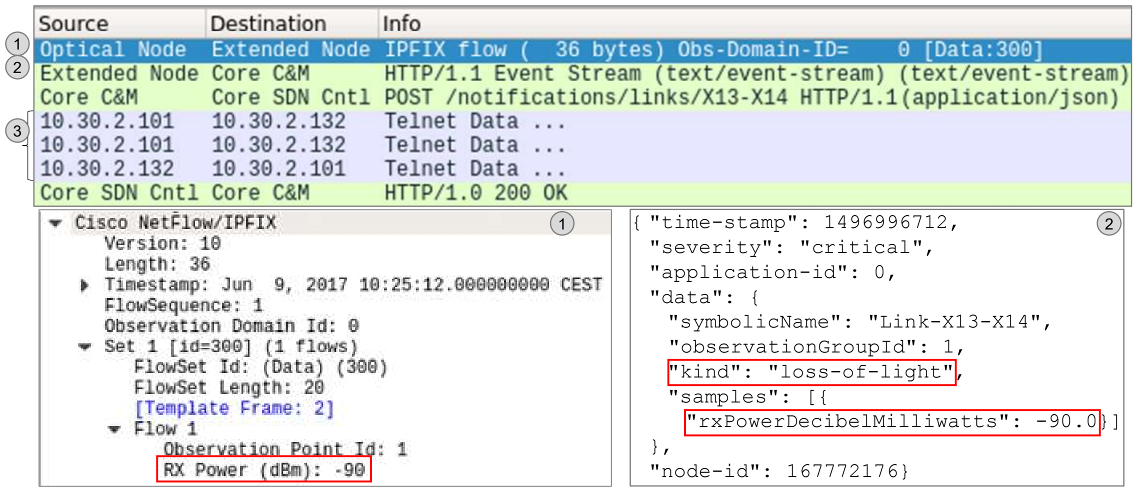


```
  Source          Destination     Info
1 Metro Node      Extended Node   IPFIX flow (  64 bytes) Obs-Domain-ID=  328 [Data:310]
2 Extended Node   Metro C&M       HTTP/1.1 200 OK  (text/event-stream)
3 Metro C&M       CNC-328         HTTP/1.1 200 OK  (text/event-stream)
  CNC-328         CNC-328 SDN Cntl POST /notifications/links/S3-S4 HTTP/1.1(application/json)
  CNC-328 SDN Cntl CNC-328        HTTP/1.1 200 OK  (text/plain)
  CNC-328 SDN Cntl CNC-328 PT     POST /pce HTTP/1.1  (application/json)
  CNC-328 PT      CNC-328 SDN Cntl HTTP/1.1 200 OK  (text/plain)
4 CNC-328 SDN Cntl Metro SDN Cntl POST /net-orchestrator/route HTTP/1.1(application/json)
5 Metro SDN Cntl  CNC-328 SDN Cntl HTTP/1.1 200 OK  (text/plain)
6 Metro SDN Cntl  Mininet         Type: OFPT_FLOW_MOD
```

```
{ "time-stamp": 1496188920,                     2
  "severity": "major",
  "application-id": 328,
  "data": {
    "symbolicName": "VLink-S3-S4",
    "observationGroupId": 1,
    "kind": "threshold-exceeded",
    "samples": [{
      "ber": 1.1700e-06,
      "rxPowerDecibelMilliwatts": -8.77}]
  },
  "node-id": 167772174}
```

```
▼ JavaScript Object Notation: application/json
  ▼ Object
    ▼ Member Key: src                            4
        String value: S1
        Key: src
    ▼ Member Key: dst
        String value: S6
        Key: dst
    ▼ Member Key: path
      ▼ Array
          String value: S1 ┐
          String value: S2 │
          String value: S5 ├ New route
          String value: S6 ┘
        Key: path
```

Fig. 8. Exchanged messages for WF1 (message numbering follows that of WF1 in Fig. 5).



```
  Source          Destination     Info
1 Optical Node    Extended Node   IPFIX flow (  36 bytes) Obs-Domain-ID=    0 [Data:300]
2 Extended Node   Core C&M        HTTP/1.1 Event Stream (text/event-stream) (text/event-stream)
  Core C&M        Core SDN Cntl   POST /notifications/links/X13-X14 HTTP/1.1(application/json)
3 10.30.2.101     10.30.2.132     Telnet Data ...
  10.30.2.101     10.30.2.132     Telnet Data ...
  10.30.2.132     10.30.2.101     Telnet Data ...
  Core SDN Cntl   Core C&M        HTTP/1.0 200 OK
```

```
▼ Cisco NetFlow/IPFIX                                  1
    Version: 10
    Length: 36
  ▶ Timestamp: Jun  9, 2017 10:25:12.000000000 CEST
    FlowSequence: 1
    Observation Domain Id: 0
  ▼ Set 1 [id=300] (1 flows)
      FlowSet Id: (Data) (300)
      FlowSet Length: 20
      [Template Frame: 2]
    ▼ Flow 1
        Observation Point Id: 1
        RX Power (dBm): -90
```

```
{ "time-stamp": 1496996712,                          2
  "severity": "critical",
  "application-id": 0,
  "data": {
    "symbolicName": "Link-X13-X14",
    "observationGroupId": 1,
    "kind": "loss-of-light",
    "samples": [{
      "rxPowerDecibelMilliwatts": -90.0}]
  },
  "node-id": 167772176}
```

Fig. 9. Exchanged messages for WF2 (message numbering follows that of WF2 in Fig. 5).

The CNC decides to reroute customer MPLS connections currently using the degraded vlink. To that end, a number of messages are exchanged afterward between the CNC, its ONOS instance and the planning tool to compute optimal rerouting; the global concurrent optimization request uses an exclude-route object (XRO) to avoid vlink S3-S4. Once the rerouting has been computed, the CNC SDN controller requests the rerouting to the metro controller and includes the computed route (message 4). Finally, metro SDN controller issues `FLOW_MOD` messages to the packet nodes.

Let us assume that at some point in time, the degraded optical link is disconnected, which triggers workflow WF2. Exchanged messages are presented in Fig. 9, where messages are numbered as in WF2 in Fig. 5. The optical power meter at the input of optical link X13-X14 detects LoL, so optical node X14 issues an IPFIX message to the extended node with such measure (message 1 in Fig. 9). Then, the extended node issues a "loss-of-light" notification with severity level "critical" to the metro controller (message 2); note that the notification specifies "application-id"=0, i.e., *Slice0*. Upon the reception of the notification, the core control and management notifies the core SDN controller, which triggers a restoration procedure for every lightpath using the failed optical link. Reconfiguration messages 3 are sent through an NETCONF proprietary API (decoded as TELNET for convenience).

Once optical link X13-X14 is restored, lightpath supporting vlink S3-S4 returns to normal BER conditions, which triggers WF3. Note that since WF3 is similar to WF1, no specific captures are presented.

VI. Conclusions

Network slicing combines virtualization and isolation to support 5G applications with ultra-low latency and high availability.

In this paper, an architecture to enable autonomic slice networking has been presented. The architecture consists of extended nodes connected to their physical counterparts for monitoring collection and data processing and analysis, thus enabling making local decisions. The domain data analytics 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.

A complex use case was defined to assess the proposed architecture. The use case encompasses three OAA loops defined in the form of workflows triggered by different events: *i*) proactive network slice rerouting after BER degradation detection in a lightpath supporting a vlink; *ii*) reactive core network restoration after optical link failure; and *iii*) reactive network slice rerouting after normal BER notification after lightpath restoration.

The proposed architecture was experimentally validated in a distributed field trial set-up connecting premises in UPC and CNIT.

ACKNOWLEDGMENTS

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 SYNERGY project (TEC2014-59995-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

[1] L. Velasco, Ll. Gifre, F. Paolucci, and F. Cugini, "First Experimental Demonstration of Autonomic Slice Networking," Post-deadline paper in OFC, 2017.

[2] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges," IEEE Communications Magazine, vol. 55, pp. 80-87, 2017.

[3] L. Velasco, L.M. Contreras, G. Ferraris, A. Stavdas, F. Cugini, M. Wiegand, and J. P. Fernández-Palacios, "A Service-Oriented Hybrid Access Network and Cloud Architecture," IEEE Communications Magazine, vol. 53, pp. 159-165, 2015.

[4] N. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, pp. 862-876, 2010.

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

[6] D. Ceccarelli and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks," IETF draft, work-in-progress, 2017.

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

[8] Ll. Gifre, A. P. Vela, M. Ruiz, J. López de Vergara, and L. Velasco, "Experimental Assessment of Node and Control Architectures to Support the Observe-Analyze-Act Loop," in Proc OFC 2017.

[9] A. P. Vela, M. Ruiz, L. Velasco, "Distributing Data Analytics for Efficient Multiple Traffic Anomalies Detection," Elsevier Computer Communications, vol. 107, pp. 1-12, 2017.

[10] A. P. Vela, M. Ruiz, F. Fresi, N. Sambo, F. Cugini, G. Meloni, L. Potì, L. Velasco, and P. Castoldi, "BER Degradation Detection and Failure Identification in Elastic Optical Networks," accepted in IEEE/OSA Journal of Lightwave Technology (JLT), 2017.

[11] L. Velasco, D. King, O. Gerstel, R. Casellas, A. Castro, and V. López, "In-Operation Network Planning," IEEE Communications Magazine, vol. 52, pp. 52-60, 2014.

[12] F. Morales, M. Ruiz, Ll. Gifre, L. M. Contreras, V. López, and L. Velasco, "Virtual Network Topology Adaptability based on Data Analytics for Traffic Prediction," (Invited) IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 9, pp. A35-A45, 2017.

[13] L. Velasco, A. P. Vela, F. Morales, and M. Ruiz, "Designing, Operating and Re-Optimizing Elastic Optical Networks," (Invited Tutorial) IEEE/OSA Journal of Lightwave Technology (JLT), vol. 35, pp. 513-526, 2017.

[14] B. Claise, A. Kobayashi, and B. Trammell, "Operation of the IP Flow Information Export (IPFIX) Protocol on IPFIX Mediators," IETF RFC 7119, 2014.

[15] gRPC Remote Procedure Call: http://www.grpc.io/

[16] Apache Cassandra: http://cassandra.apache.org/

[17] Apache Spark: http://spark.apache.org/

[18] A. Castro, R. Martínez, R. Casellas, L. Velasco, R. Muñoz, R. Vilalta, and J. Comellas, "Experimental Assessment of Bulk Path Restoration in Multi-layer Networks using PCE-based Global Concurrent Optimization," IEEE J. of Lightwave Techn., vol. 32, pp. 81-90, 2014.

[19] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in Proc ACM SIGCOMM HOTNETS, 2010.

[20] ONOS (Open Network Operating System). [Online] http://www.onosproject.org/ [Last accessed: Jun. 2017].

[21] P. Pavon-Marino and J.-L. Izquierdo-Zaragoza, "Net2Plan: An Open Source Network Planning Tool for Bridging the Gap between Academia and Industry," IEEE Network, vol. 29, pp. 90–96, 2015.

[22] F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "Service Chaining in Multi-Layer Networks using Segment Routing and Extended BGP FlowSpec", in Proc. OFC 2017.

[23] M. Dallaglio, N. Sambo, F. Cugini, and P. Castoldi, "Control and Management of Transponders with NETCONF and YANG," IEEE/OSA Journal of Optical Communications and Networking, vol. 9, pp. B43-B52, 2017.