Dynamic Monitoring of Data Center Slices

Francesco Tusa, Stuart Clayman, Alex Galis
Dept. of Electronic Engineering, University College London, London, UK
Email: francesco.tusa@ucl.ac.uk, s.clayman@ucl.ac.uk, a.galis@ucl.ac.uk

***Abstract*—Slicing is a move towards segmentation of resources and deployment of NFV for the purpose of enhanced services and applications on globally shared resources. The slicing approach in this paper considers Data Center slicing and the VIM on-demand model. We focus on the monitoring of Data Center slices, showing what is needed from the monitoring perspective and how the monitoring should be done. The proposed monitoring approach is validated on a platform that supports the on-demand creation of lightweight VIM instances.**

***Index Terms*—Cloud monitoring, resource monitoring, slice monitoring, monitoring abstraction, dynamic control, on-demand reconfiguration, network service orchestration, NFV, SDN, 5G.**

I. Introduction

Slicing is a move towards segmentation of resources and deployment of NFV for the purpose of enhanced services and applications on globally shared resources. In order to support service provisioning over a slice-enabled distributed NFVI (for NFV Infrastructure), new mechanisms that implement slicing over the whole end-to-end infrastructure – from the mobile edge to the core DC – including network, compute and storage resources need to be implemented for sliced services.

A slice-based orchestration approach will provide a more effective resource management due to the isolation introduced in both the control and data planes. A Slice will represent a self-contained bundle of heterogeneous resources accessible via a management and control interface that can be utilised to dynamically modify both the slice topology and configuration at run-time, according to the overall status of the infrastructure.

The slicing approach being considered in this paper aims at bridging the conceptual separation between a pure network slice and a Data Center (DC) slice in order to create an end-to-end slice that encompasses the different segments of a multi-provider NFVI. To design this new slicing approach, we have made the case for creating a VIM (Virtual Infrastructure Manager) on-demand and dynamically allocating a new VIM for each DC slice segment, rather than having one VIM for the whole DC [10] [5].

A Slicing Orchestrator with dedicated components, in the Control Plane, will drive the creation of the slices, will take care of the slice life-cycle management, and will also include the dynamic slice reconfiguration based on elasticity rules. We observe on the one hand the slice management interface will enable abstract, unified management and control of the resources constituting a slice; on the other hand, the monitoring process becomes a fundamental aspect to be considered when implementing slicing, as it will be used to provide feedback about the slice utilisation. This will allow the Slicing Orchestration layers to enforce dynamic slice elasticity, reconfiguration, and so on.

978-1-5386-9376-6/19/$31.00

In this paper we will focus on the monitoring aspects of DC slicing. We consider a proper way to collect the status of the DC slice elements and also to expose that information to the Slicing Orchestration layer, regardless of both the resource type (computing, storage and connectivity) and technological implementation. For this reason, we introduced a Slice Monitoring Abstraction (SMA) layer between the Slicing Orchestration functions and the slice's resource infrastructure. This is required to support and complement the abstraction characteristics of a slice. The SMA will provide the proper abstractions on top of the resources bundled in each slice by creating an on-demand, bespoke instance of a monitoring system that follows the slice's initiation pattern. The monitoring elements allocated by the SMA for each slice will be tailored to the specific resources of that slice, and will allow a Slice Orchestrator to collect information about the slice status and utilisation in a uniform way.

The SMA layer presented in this paper utilises the Lattice Monitoring Framework [3]. Lattice was designed for highly dynamic and virtualised environments, and has recently been augmented with mechanisms to support on-demand deployment and dynamic reconfiguration of its monitoring elements [11] in order to build a monitoring subsystem on-the-fly via software control – Software Defined Infrastructure Monitoring. In this paper we will explain how those features have been utilised in the context of a sliced end-to-end infrastructure to support the abstraction requirements of the slices, and to instantiate a bespoke slice monitoring adaptor for each slice at run-time.

II. Related Works and Background

The goal of providing services on top of slices, i.e., logically partitioned resources of a multi-technology, multi-domain software-defined NFVI, can be obtained through different slicing strategies according to which specific system elements provide the slicing and at what layer the slicing is introduced.

When slicing is implemented at the lower layers, namely the infrastructure, the upper layers, such as VIMs and Orchestrators, do not need to be slicing-aware. If a slice is presented to them, they can carry on working with minimal or even no change. Conversely, if slicing is done in the Orchestrator, which uses an inter-domain orchestration API interaction and / or a peer to peer approach, a slice becomes closer to a set of data structures in the Orchestrator rather than

an actual resource partition. Consequently, there are inherent trade-offs when selecting one or the other slicing approach. The actual decision on which slicing approach should be used will depend on various key aspects of the service requirements under consideration, and can be focused on the technical desires of the provider, together with the technical abilities and technological choices of the tenants.

There are various standards organisations that have been addressing slicing and creating various definitions (ITU-T IMT2010/SG13, ETSI, 3GPP TR23.799, ONF TR-526, IETF and IEEE) as well as EU 5G-PPP White Papers on 5G Architecture centred on network slicing (mark 1 and 2) and various projects and initiatives that are considering slicing implemented at the Orchestrator level. (SONATA, 5GEx, 5G Transformer, 5G!Pagoda and SLICENET).

Monitoring solutions that were already designed for monitoring traditional IT infrastructures are not suitable for slice-enabled software-defined NFVI scenarios as they lack some relevant desired features: namely the ability to seamlessly support different technologies, attaching / detaching new monitoring elements to the system at run-time in a software-defined fashion, and re-configuring their behaviour without restarting the running components.

The Distributed Architecture for Resource manaGement and mOnitoring in cloudS (DARGOS) [1] is a Cloud monitoring architecture to disseminate resource monitoring information that is flexible, adaptable and allows defining and monitoring new metrics easily. This solution has some commonalities with Lattice as it cares about optimising the network traffic by keeping a low overhead while the measurements are sent. However, DARGOS does not provide any control element nor an API for programming and automating either the deployment and (re)-configuration of the monitoring subsystem. FlexACMS [6] is a framework to automate monitoring configuration related to cloud slices using multiple monitoring solutions. FlexACMS is able to detect new cloud slices created in the cloud platform and, for each new detected cloud slice, to trigger components that can configure the monitoring solutions, building the corresponding monitoring slice for that cloud slice.

Prometheus [8] is an open-source systems monitoring and alerting toolkit that joined the Cloud Native Computing Foundation in 2016. It works well for recording any purely numeric time series and fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architectures. Although Prometheus offers a more dynamic approach compared to other monitoring solutions, its architecture is still conceived around the concept of a central server entity that needs to be configured and restarted every time a configuration change / adjustment is required.

III. Data Center Slices

When looking at the current initiatives that are dealing with slicing, we observe that slices can be requested from networks and combined with various service elements plus VNFs (which of course run in Data Centers), which are then presented in the form of a *Network Slice*. This is currently on-going work in many organisations such as ITU, IETF and the IEEE. However, although networks and Data Centers are physically connected, we identified an anomaly in the fact that it is not usually possible to request a slice from a Data Center to build a distributed end-to-end slice that includes an ad-hoc partitioned set of computation, storage, and network resources to support the deployment of network services. In this work, we focus on the concept that slices should also be a feature that can be requested from Data Centers to be then combined with the network slice parts, in order to build full end-to-end slices.

A Data Center (DC) slice, which from now on will be the focus of this work, is an abstraction over the resources of a DC, and provides a mechanism to manifest infrastructure slicing, such that:

- a DC slice presents a collection of resources that look like a DC, only smaller, and
- a DC slice can be controlled, managed, and monitored independently from any other DC slices.

For each DC Slice requested, there will be a VIM allocated on-demand to service that DC Slice, as outlined in [5]. This VIM will be only for that one slice, and will be independent of the VIMs running in the rest of the Data Center. Given that these on-demand VIMs for DC Slices are not pre-existent, they can be allocated for any kind of lower level virtualization, including Xen and KVM; or for containers such as Docker and Kubernetes. As a consequence, this choice and flexibility is not a feature pre-determined once by the DC or the provider, but can now be an option for the customer. Furthermore, as the VIM is allocated for the slice and is independent from other VIMs, the customer can have a lot more ability to adjust the configuration options for their VIM. It also means that the customer could also be billed for their VIM, as opposed to it being part of the shared infrastructure.

Once the different DC slices have been allocated and separate independent VIMs have been spawned therein, then a full end-to-end slice can be built by stitching together the different DC Slice segments via the relevant network slice parts. For each created end-to-end slice, it is necessary to build a uniform, on-demand monitoring layer to aggregate monitoring measurements from the separate DC Slice segments, regardless of the underlying VIM technology being used.

It is this uniform, on-demand monitoring layer that is the focus here, and we address the way DC Slices are monitored and controlled, following an on-demand dynamic allocation pattern. In order for the existing orchestration layers (e.g., NFV MANO) to carry out their functions in such a slice-enabled environment, a monitoring abstraction layer must be deployed on top of the end-to-end infrastructure, and should follow the slice creation pattern. That is, different VIMs can be used for different slice parts, and the slices can be created an destroyed in a dynamic way according to the network service instances requested by the tenants.

IV. Lattice Monitoring Framework

Lattice was conceived as a highly flexible and versatile framework able to provide adaptable building blocks for

building monitoring systems [3], and was designed and implemented around the concept of elements that can be dynamically composed as required, where the data collection task is performed by *probes* able to retrieve measurements from different types of sources. A probe collects measurements related to the elements of the system that needs to be monitored, and provides a uniform way to collect the measurements regardless of the specific type of entity to be monitored.

The Lattice framework is well suited to environments where a network service or a resource bundle / slice that needs to be monitored is based on a heterogeneous combination of compute, storage and network resources allocated either in a physical or virtualised form. Different implementations of probes can be used to interact with different types of objects that may be part of a slice, or that are dynamically created and destroyed therein, eventually providing uniform types of measurements as outcome of the overall monitoring process. The usage of a monitoring subsystem based on Lattice can thus provide an abstract way to collect different measurements from the heterogeneous types of resources that can be bundled within a slice (see [2], [3] and [11] for further details). In this paper we will consider the DC Slice monitoring only.

Lattice has recently been extended to support mechanisms that enable the dynamic instantiation, configuration and control of all the monitoring entities that are going to be part of a Lattice-based monitoring subsystem. This potentially allows the deployment of a bespoke monitoring subsystem on-demand as well as to perform its dynamic reconfiguration and adaptation according to the status of the main system that needs to be monitored (e.g., the status of resources and services, the overall number of instantiated entities, the instantiated slices and their status, etc.). The Lattice Control and Information planes [11] play a fundamental role in the implementation of those functionalities.

The Lattice Information and Control planes are part of the design specification described in [2] and [3], have fully been implemented in the latest version of Lattice (using the ZeroMQ asynchronous messaging library [7]) described in [11]. The Information plane is used to share a topological view of the monitoring entities that are up and running in the monitoring subsystem (Probes and Data Sources as well as Data Consumers and Reporters), also including their properties / attributes and run-time status. The Control plane is utilised to enable one, or more, Monitoring Controller entities to interact with the other Lattice monitoring elements that are under their control in order to enforce control actions (e.g., activating / deactivating a given probe or a reporting mechanism, dynamically adjusting the measurements rate of a probe, etc.).

The Lattice Monitoring Controller has fully been implemented and added to the other existing components of the Lattice framework. The Monitoring Controller supports the following dynamic run-time functionalities: (i) Starting / Stopping a Data Source monitoring agent on host / resource; (ii) Loading / Unloading a probe (on a Data Source) with attributes (service id, NF id, Virtual Link id, etc.); (iii) Activating / Deactivating a probe; (iv) Setting a probe measurements collection / transmission rate; (v) Starting / Stopping a Data Consumer monitoring agent on host / resource; (vi) Loading / Unloading a reporter (on a Data Consumer monitoring agent); and (vii) Configuring a reporter to use a particular persistent storage system for storing the collected measurements.

These functions are exposed by the Monitoring Controller via a RESTful API, which triggers proper control messages accordingly. The control messages implement a request-reply protocol involving the Monitoring Controller and the Monitoring elements that need to be controlled. A lightweight RPC mechanism was implemented using the XDR (External Data Representation) [9] to enforce the remote control of whatever running entity that requires dynamic control / reconfiguration via triggering the execution of proper methods / procedures. The result of each control operation is then passed back to the Monitoring Controller on the Control Plane and used as return value of the original REST API call.

These mechanisms become fundamental to support the monitoring abstractions within the SMA layer as they can enable dynamic instantiation / deployment and control of the monitoring elements of each end-to-end slice. Using the Lattice dynamic management and control features, probes can be loaded on specific Lattice Data Sources according to the "spatial" configuration of the end-to-end slices being instantiated and most importantly considering the type of technology of each slice segment (e.g., the type of running VIM in a DC Slice).

V. Design for Monitoring of DC Slices

A monitoring sub-system that supports the deployment and configuration of all the required monitoring entities at run-time, as well as their dynamic control is essential, given the context of slices. This will allow the dynamic instantiation of the monitoring abstractions required for each particular slice, as well as the ability to adapt the deployment, configuration, and behaviour of the Slice Monitoring Adaptation (SMA) according to the run-time status of the system.

For doing that, the implementation of the SMA should be based on a monitoring subsystem that (i) is able to collect and provide measurements from different types of resources, regardless of the technology used for their implementation; (ii) is able to easily scale up / down according to the number of running entities in the system as result of the instantiation / termination of multiple services / slice segments; (iii) provides mechanisms to dynamically activate / deactivate its constituent elements (e.g., probes) on demand according to the type of services / resources to be monitored; (iv) is able to provide mechanisms to dynamically adjust the configuration if its elements, e.g., the measurements collection / sending rate according to the status of the infrastructure, the number of running entities and the characteristics of the slices (e.g., defined thresholds and events, SLAs, etc.).

Consider a single Data Center on which multiple DC Slices segments are allocated. The Data Center supports multiple

independent DC slices which are managed by remote Orchestrators to ensure modularity, isolation, and security [5]. During the allocation of a new DC Slice segment, a Slice Orchestrator also allocates the required monitoring adaptation layer which provides a uniform measurements view of each Slice segment, regardless of the implementation and the particular type of deployed VIM. The representation, shown in Figure 1, is focused on the slice allocation, which is performed by a Slice Orchestrator function embedded in the Orchestration Layer, interacting with the DC Slice Controller of the Data Center.

When an *end-to-end slice* is created by the Slice Orchestrator, different DC Slices (segments) are allocated in different Data centers, each possibly using a different type of VIM. As the Slice Orchestrator is aware of the particular type of VIM used in each DC Slice, it can interact with the Slice Monitoring Abstraction (SMA) via its *control interface* to trigger the on-demand instantiation of a Slice Monitoring Adapter for that particular slice and VIM type. Figure 2 highlights the way that the SMA has been implemented using the features and components provided by Lattice. The Monitoring Controller within the SMA will take care of translating the monitoring requests sent by the Slice Orchestrator into the instantiation of the monitoring components required to collect measurements from each DC Slice segment. This will be done on-demand by allocating and configuring one of more Lattice Data Source elements associated to a particular DC Slice together with one or more Probes that will collect relevant measurements for that segment of the end-to-end slice.

Each of those probes will interact with the particular VIM and / or monitoring system already available in the DC Slice, and will hide the related implementation details to the Slice

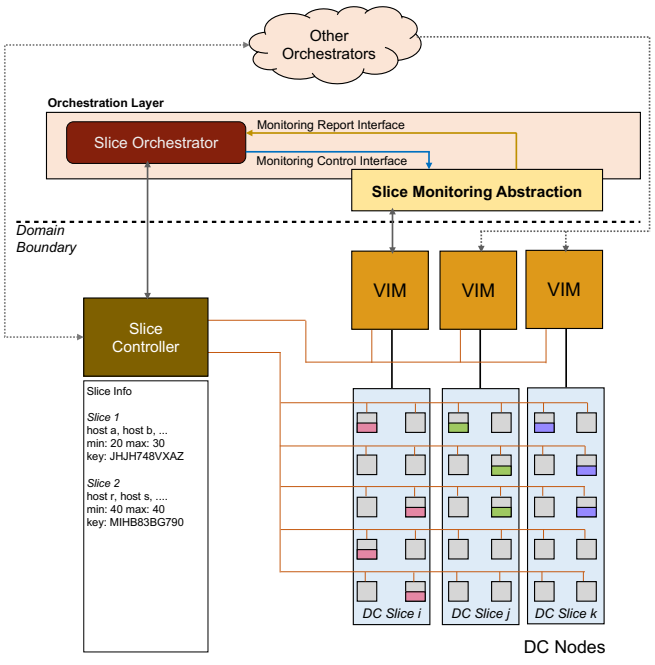

Figure 1: A Sliced Data Center

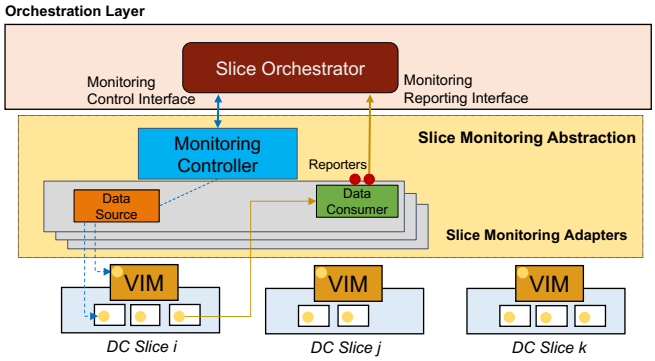

Figure 2: SMA Implementation using Lattice

Orchestrator. The measurements collected from a given DC slice by the on-demand instantiated probes, will be sent to an aggregation element implemented via a Lattice Data Consumer, which will take care of formatting them using a uniform data model before passing them on to the Slice Orchestrator to be consumed.

The Slice Orchestrator will be provided with an overall and uniform view of the status of the resource (both physical and virtual) that have been assigned to each end-to-end slice, as the above approach will be performed for collecting monitoring measurements from all the segments of each end-to-end slice. A global view of the end-to-end resource infrastructure will be built using the monitoring information from all the end-to-end slices and exploited by the Slice Orchestrator as feedback for the execution of its services and functions.

VI. Evaluation

In order to prove the effectiveness of the monitoring abstraction layer described in this paper, we have built a fully integrated Multi MANO platform, presented in [12], using Lattice. The platform has end-to-end slices which consist of the composition of lightweight DC slices and Network slices (out of the scope of this paper), where each DC slice runs an instance of the VLSP (Very Lightweight Network & Service Platform) VIM [4]. When the creation of an end-to-end slice is triggered and performed by the Slice Orchestrator, a monitoring activation request is also propagated to the SMA component via the Control Interface, and a new Slice Monitoring Adaptor is instantiated on-demand for each DC slice segment. Then, one or more Lattice Data Source objects are allocated in each slice (depending on the number of elements to be monitored), together with the specific type of *probes* that allow collecting information from the physical and virtual resources of that slice segment. For each slice segment, and each Slice Monitoring Adaptor, the measurements collected by the *probes* and sent by the Lattice Data Sources are aggregated by a Lattice Data Consumer, which is also configured at the moment of creation of the Slice Monitoring Adaptor to report the aggregated measurements back to the Slice Orchestrator, via the Reporting Interface represented in Figures 1 and 2.

Each slice segment will run an instance of the lightweight VLSP VIM, all the instantiated probes will implement the required mechanisms to collect monitoring information via the VLSP REST monitoring API exposed by its Global Controller. The information will include both the resource utilisation of the physical hosts assigned to the DC Slice as well as the status of the allocated virtual objects. All the measurements will properly be structured with respect to the Lattice data model and each of the measurement objects will in turn be encoded using the XDR format [9], before being sent over the Data Plane. The XDR encoding approach is used in order to reduce the amount of network traffic used by the streams of monitoring data. XDR is a standard data serialization format which allows data to be transferred between different kinds of computer systems, and became an IETF standard in 1995. Figure 3 provides an example of aggregated measurements of a VLSP DC Slice sent by a Data Consumer to the Slice Orchestrator via the Reporting Interface.

```
Host ID: 1
    cpuLoad: 17.83          cpuIdle: 82.16
    usedMemory: 2.140625    freeMemory: 1.8007812
    energyTot: 3.416218     energyDelta: 0.568652
    energyNow: 0.568652

        id: Router-4
            cpuLoad: 141.164        cpuIdle: 1188.411
            usedMemory: 1005.539    freeMemory: 182.872
            energyTot: 50682.0      energyDelta: 2.318565
            energyNow: 0.14787656

        id: Router-5
            cpuLoad: 131.142        cpuIdle: 802.48
            usedMemory: 666.287     freeMemory: 136.193
            energyTot: 30214.0      energyDelta: 1.5388933
            energyNow: 0.09207494
```

Figure 3: Aggregated measurements from a DC Slice

Although in the Multi MANO platform, all of the slice segments consisted of lightweight VLSP DC Slices, when a different virtualization technology is in use, the same approach of using a Slice Monitoring Adaptor, allocated on-demand for each VIM type, such as Openstack, Docker, etc., and can be applied for collecting and encoding the measurements by using the specific probe implementation for the type of VIM running in the slice segment, or for monitoring systems that might already be installed and are up and running in the DC slice, such as Ceilometer, Prometheus, etc.

VII. Conclusions and Future works

This paper is focused on a slicing approach implemented at the infrastructural layer using the concept of Data Center slicing coupled with VIM on-demand – this allows achieving a slicing approach which remains transparent to the upper layers and Orchestrators operating on top of the NFVI. In particular we focused on the monitoring of these Data Center slices, showing what is needed from the monitoring system and how the monitoring of those slices is done.

In this dynamic environment, it is fundamental that the slice monitoring mechanisms are implemented following these dynamic resource allocation patterns. For this reason, we considered in this paper the research problem related to the design and implementation of such a dynamic monitoring layer, and we provided a first solution based on a new component that we named *Slice Monitoring Abstraction (SMA)*, which is based on the Lattice Monitoring Framework.

To demonstrate its feasiblity and operation, the implementation of the designed abstractions, APIs and data models were integrated and functionally evaluated in that real platform where end-to-end slices including lightweight VLSP DC segments were instantiated and monitored in a uniform way via the on-demand deployment of different Slice Monitoring Adaptors performed via the SMA. A snapshot of the monitoring data collected was presented.

In future works, we plan to extend the above testing scenario with the instantiation of end-to-end slices whose DC Slice segments include different technologies and VIMs in order to further prove the effectiveness of our proposed approach. Furthermore, work will be aimed at identifying the best approach to monitor the Network slice parts of an end-to-end slice (which was not in the scope if this paper), and extending the data models and abstractions proposed here in order to achieve a unified approach for monitoring both the DC and Network Slice segments that are part of an end-to-end Slice.

Acknowledgements

This work was partially supported by the EU-Brazil project: NECOS – “Novel Enablers for Cloud Slicing” (777067).

References

[1] DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds. *Future Generation Computer Systems*, 29(8):2041 – 2056, 2013.

[2] S. Clayman, A. Galis, C. Chapman, G. Toffetti, L. Rodero-Merino, L.M. Vaquero, K. Nagin, and B. Rochwerger. Monitoring Service Clouds in the Future Internet. In *Towards the Future Internet - Emerging Trends from European Research*. IOS Press, Fourth 2010.

[3] S. Clayman, A. Galis, and L. Mamatas. Monitoring virtual networks with Lattice. In *2010 IEEE/IFIP Network Operations and Management Symposium Workshops*, pages 239–246, April 2010.

[4] S. Clayman, L. Mamatas, and A. Galis. Experimenting with control operations in Software Defined Infrastructures. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 390–396, June 2016.

[5] S. Clayman, F. Tusa, and A. Galis. Extending Slices into Data Centers: the VIM on-demand model. In *9th International Conference on the Network of the Future (NOF)*, pages 31–38, Nov 2018.

[6] M. B. de Carvalho, R. P. Esteves, G. da Cunha Rodrigues, C. C. Marquezan, L. Z. Granville, and L. M. R. Tarouco. Efficient configuration of monitoring slices for cloud platform administrators. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, June 2014.

[7] iMatix Corporation. Zero MQ Distributed Messaging, 2019. URL: http://zeromq.org.

[8] Prometheus. Prometheus – monitoring system and time-series database, 2019 (accessed on 2019-01-07). URL: https://prometheus.io.

[9] R. Srinivasan. XDR: eXternal Data Representation standard, 1995.

[10] Stuart Clayman. Network Slicing Supported by Dynamic VIM Instantiation. In *NFVRG, IETF 100, Singapore*, 2017.

[11] F. Tusa, S. Clayman, and A. Galis. Real-time management and control of monitoring elements in dynamic cloud network systems. In *IEEE 7th International Conference on Cloud Networking (CloudNet)*, 2018.

[12] Francesco Tusa, Stuart Clayman, Dario Valocchi, and Alex Galis. Multi-Domain Orchestration for the Deployment and Management of Services on a Slice Enabled NFVI. In *IEEE NFV-SDN - Mobislice*, Verona, Italy, November 2018.