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 that of Data center slicing and the VIM on-demand model. In particular we focus on the monitoring of these Data Center slices, showing what is needed from the monitoring system and how the monitoring of slices is done. To demonstrate its feasiblity and operation we discuss how we fully integrated this approach into a Multi MANO (Management and Orchestration) platform.**

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

I. Introduction

5G Network Service Providers have been reshaping their own resource infrastructures, which are gradually evolving from a static and quite unyielding design to a new more flexible and dynamic software-based approach – Software Defined Infrastructures, i.e., a combination of NFV, SDN and Cloud technologies designed to fulfil customer demands via relatively complex software systems that perform the end-to-end orchestration of network services on heterogeneous resources, even across administrative borders.

In this multi-technology, multi-domain scenario, logically inter-connected virtualised resources are dynamically allocated / deallocated on-demand following the pattern of the network services requested by different tenants. Services usually include compute, storage, and network resources that are allocated on the global physical infrastructure as a combination of virtual machines (VMs), Containers, virtual storage devices, and virtual links.

5G network infrastructures have been conceived to support different use-cases as advised by [17], where applications and services related to each use-case can possibly introduce different orthogonal requirements in terms of expected KPIs and SLAs. In order to deal with such heterogeneity and to support the co-existence of multiple diverse service instances belonging to different tenants on the same end-to-end infrastructure, resource slicing is emerging as a promising approach to guarantee resource isolation.

Slicing is in fact 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 will definitely need to be implemented.

The 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 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 slice segment, rather than having one VIM for the whole DC [23] [8].

Dedicated Slicing Orchestrator components in the Control Plane will drive the creation of the slices and will take care of their life-cycle management, which will also include the slice dynamic reconfiguration based on elasticity rules, etc. This will happen by accessing each slice's exposed management interface, which allows enforcing operations on the aggregated slice resources regardless of their specific implementation (e.g., the type of VIM in a slice segment, etc.).

We observe on the one hand the slice 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 in order to allow the Slicing Orchestration layers to enforce dynamic slice elasticity, reconfiguration, etc.

In this paper we consider slice interface and will focus on those aspects of slicing that are more related to the monitoring of resources. In particular, a proper way to collect the status of the slice resources and to expose that information to the Slicing Orchestration layer regardless of the resource type (computing, storage and connectivity) and technological implementation, is certainly required to support and complement the abstraction characteristics of a slice.

For this reason, we introduced a Slice Monitoring Abstraction (SMA) layer between the Slicing Orchestration functions and the resource infrastructure. 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, as the slices are requested and allocated. The monitoring adaptation 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 slice utilisation in a uniform way, regardless of the slice implementation.

The SMA layer presented in this paper utilizes the Lattice Monitoring Framework [6]. Lattice was designed for highly dynamic and virtualized environemts, and has recently been augmented to support on-demand deployment and reconfiguration of its monitoring elements [24] in order to build a dynamic monitoring subsystem on-demand. 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-domain Software Defined Infrastructure (SDI), 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 orchestrator 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 focussed 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. The ITU-T Slicing model is defined in [16], and is the basic concept of the Network Softwarization. Slicing has also been addressed in ITU-T IMT2010/SG13 [15]. There is the ETSI Report on Net Slicing Support within the ETSI NFV Architecture Framework [10], the 3GPP TR23.799 Study Item on "Study on Architecture for Next Generation System" [1] which addresses Network Slicing, and the ONF Recommendation TR-526 "Applying SDN architecture to Network Slicing". The IETF Network Slicing - Revised Problem Statement [13] has various documents on Architecture, Management, Use-Cases, the Information Model, Autonomics, Gateway functions, and a Framework for Abstraction and Control of Traffic Engineered Networks (ACTN). There are the EU 5G-PPP White Papers on 5G Architecture centred on network slicing (mark 1 - (2016)) [11]) (mark 2 - (2018)) [12]. There are also currently various projects and initiatives that are considering slicing implemented at the Orchestrator level. These include SONATA [21], 5GEx [3], 5G Transformer [2], 5G!Pagoda [4] and SLICENET [20].

As this work considers the slicing approach based at the *infrastructure level*, we present a brief overview of monitoring solutions that might be suitable to this context. Our considered slicing approach based on the VIM on-demand, poses some challenges on the implementation of the monitoring layer: the slice monitoring will need to provide measurements collected from each particular VIM instance running in the different segments of the end-to-end slice in a uniform and abstracted way. Moreover, this monitoring layer will have to be deployed and configured on-demand, coherently with the resources that are bundled within each slice. A slice is in fact created dynamically according to the type and number of services requested by the tenants, and consequently it will not be possible making any a-priori assumptions on the monitoring technologies that will have to be used for a given slice. The SMA layer that we introduced will be responsible to identify at run-time the right monitoring approach to be used every time a slice is defined and allocated.

Our proposed implementation of the SMA is based on the Lattice Monitoring framework as it already provides the right mechanisms required to build such a monitoring adaptation layer on demand (See Section III). In this section we justify our choice by analysing similar features offered by other monitoring solutions and identifying functional gaps when compared to Lattice.

Although the approach to the design of software solutions for monitoring distributed resources and services has gradually evolved to support higher degrees of flexibility and dynamism, with the rise of Cloud computing, we believe that solutions already designed for monitoring traditional IT infrastructures are not suitable for slice-enabled SDI scenarios. Their main limitations are in the lack of some required features, e.g., seamlessly supporting different technologies, attaching / detaching new monitoring elements to the system at run time in a software-defined fashion, or re-configuring their behaviour without restarting their running components.

Distributed Architecture for Resource manaGement and mOnitoring in cloudS (DARGOS) [18] is a completely distributed and highly efficient Cloud monitoring architecture to disseminate resource monitoring information. DARGOS ensures an accurate measurement of physical and virtual resources in the Cloud keeping at the same time a low overhead. In addition, DARGOS is flexible and 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 [9] 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. Although this solution represents a huge steps forward in respect to the automatic deployment and configuration of a monitoring subsystem, it still does not consider any on-demand adaptation e.g., according to the conditions of the system to be monitored, the number of running entities, etc.

Prometheus [19] is an open-source systems monitoring and alerting toolkit originally built at SoundCloud that has recently become a standalone open source project as it joined the Cloud Native Computing Foundation in 2016 (it is now the second hosted project after Kubernetes). Prometheus works well for recording any purely numeric time series. It fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architectures. Each Prometheus server is standalone, not depending on network storage or other remote services. It is possible relying on it when other parts the infrastructure are broken, and it is not needed setting up extensive infrastructure to use it. 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 adjusted every time a configuration change is required.

III. Lattice Monitoring Framework

In its original design and developed software prototype, Lattice was conceived to be a highly flexible and versatile framework able to provide the software blocks that can be used to build an ad-hoc monitoring subsystem [6].

Lattice was designed and implemented around the concept of software elements that can dynamically be composed as required, and where the data collection task is performed by *monitoring probes*, i.e., Java software modules that can be dynamically programmed and used to retrieve measurements from different types of sources (see Figure 1). A probe usually collects measurements related to the elements of the system that needs to be monitored. To add more flexibility when building the required monitoring subsystem, Lattice allows performing the measurement collection process using two alternative approaches: interacting either with those elements directly (e.g., a physical host, a hypervisor, a network controller, an OVS switch, etc.) or with an existing higher level system that manages / controls / monitors those entities already (e.g., a Docker Engine, an OpenStack Ceilometer, a Prometheus DB, an SDN Controller, etc.). In both cases, the role of the probe will be that of providing the proper interaction mechanisms for that source, and to provide a uniform way to collect the measurements regardless of the specific type of entity to be monitored.

For this reason, the Lattice framework well suits 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 already 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.

Probes are not the only elements provided by the Lattice framework in order to build a tailor-made monitoring subsystem: the current specification of Lattice also includes *Data Sources*, *Data Consumers* and *Reporters*, where each of them implements specific functions required for the monitoring process (Figure 1, see [5], [6] and [24] for further details).

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 [24] will play a fundamental role in the implementation of those functionalities.

A. Requirements for a Monitoring Subsystem in a Sliced infrastructure

A monitoring subsystem that supports the deployment and configuration of all the required monitoring entities at run-time, as well as their dynamic control would be desirable to have in the context discussed in the previous section. This would allow to dynamically instantiate the monitoring abstractions required for each particular slice as well as to reshape the deployment, configuration and behaviour of the Slice Monitoring Adaptation (SMA) layer 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; (iii) provides mechanisms to dynamically activate / deactivate its constituent elements (e.g., probes) on demand according to the type of services to be

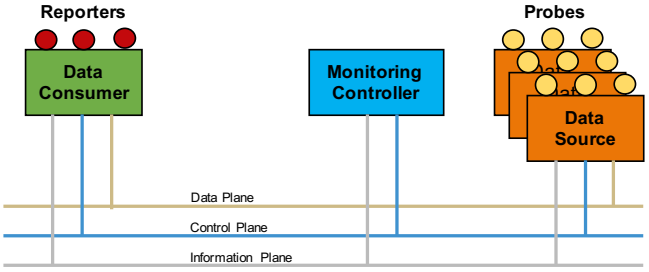

Figure 1: Lattice Framework Elements

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.).

B. Dynamic control features

The Lattice Control and Information planes that were included as part of the design specification described in [5] and [6], have fully been implemented (using the ZeroMQ asynchronous messaging library [14]) in the latest version of Lattice described in [24].

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 (represented in Figure 1) 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.).

Recently, the Lattice Monitoring Controller has fully been designed, implemented and added to the other existing components of the Lattice framework. The Monitoring Controller supports the following dynamic run-time functionalities:

- Starting / Stopping a Data Source monitoring agent on host / resource,
- Loading / Unloading a probe (on a Data Source) with attributes (service id, NF id, Virtual Link id, etc.),
- Activating / Deactivating a probe,
- Setting a probe measurements collection / transmission rate,
- Starting / Stopping a Data Consumer monitoring agent on host / resource,
- Loading / Unloading a reporter (on a Data Consumer monitoring agent),
- 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 that includes different endpoints. The API invocation on the Controller will first trigger the generation and then the transmission of proper control messages on the Lattice Control Plane. These messages implement a request-reply protocol involving the Monitoring Controller and the Monitoring elements that need to be controlled. More specifically, a lightweight RPC mechanism was implemented using the External Data Representation (XDR) to allow enforcing 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 turned out to be fundamental in support of the monitoring abstractions within the SMA layer as they enabled dynamic control features for the main monitoring elements of each slice. Using the Lattice dynamic management and control features, probes can be loaded on specific Data Sources monitoring elements according to the "spatial" configuration of the slices being instantiated and most importantly considering the type of technology of each slice segment (e.g., the type of running VIM).

Although Lattice can be used to build the monitoring functionalities of any kind of dynamic system requiring on-demand management and control, in the next section we will specifically consider a 5G multi-provider sliced scenario where a Lattice-based Slicing Monitoring Abstraction layer was devised to support the dynamic monitoring of Data Center Slices. We will discuss and highlight how some of the Lattice features turned out to be pretty useful to support on-demand deployment, control and adaptation features for that system.

IV. Data Center Slices

When looking at the current initiatives that are dealing with slicing, we observed that slices can be requested from networks and combined with various service elements plus NFV and then are presented in the form of a *Network Slice*. This is currently an on-going work in many organisations such as ITU, IETF and the IEEE. However, although Data Centers and the networks 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. 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 slicing parts in order to build end-to-end slices.

A Data Center (DC) slice 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 and managed independently from any other DC slices.

For each DC Slice requested, there will be a VIM allocated on-demand to service that DC Slice. This VIM will be only for that one slice, and will be independent of the VIM managing 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, as outlined in [8], then a full end-to-end slice can be built by stitching together the different DC Slice segments via the proper network slicing parts. For each created end-to-end slice, it would be required 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 research problem we try to solve in this paper, and we focus on the way DC Slices are monitored and controlled on-demand, following a dynamic allocation pattern. In order for the management and orchestration layers to carry out their functions in such a slice-enabled environment, a monitoring abstraction layer should be deployed on top of the Infrastructure Managers of the end-to-end infrastructure, and should follow the slice creation pattern. 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.

V. Dynamic Monitoring of Data Center Slices

In this work we consider a single 5G Management and Orchestration (MANO) instance (in pink) operating over resources of a single Data Center on which multiple DC Slices are allocated, shown in Figure 2. The representation is focused on the slice allocation, which is performed by a Slice Orchestrator embedded in the 5G Orchestrator, interacting with the DC Slice Controller of the Data Center. The Data Center supports multiple independent DC slices which are

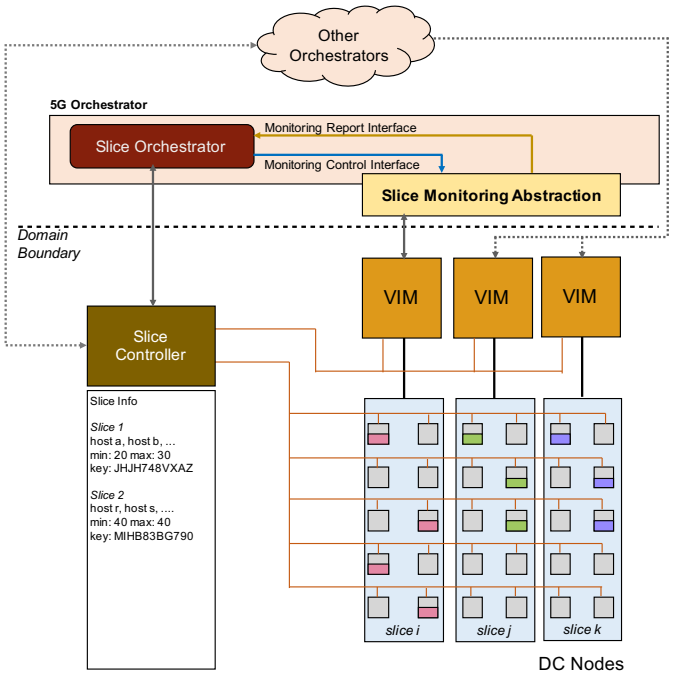

Figure 2: A 5G Sliced Data Center

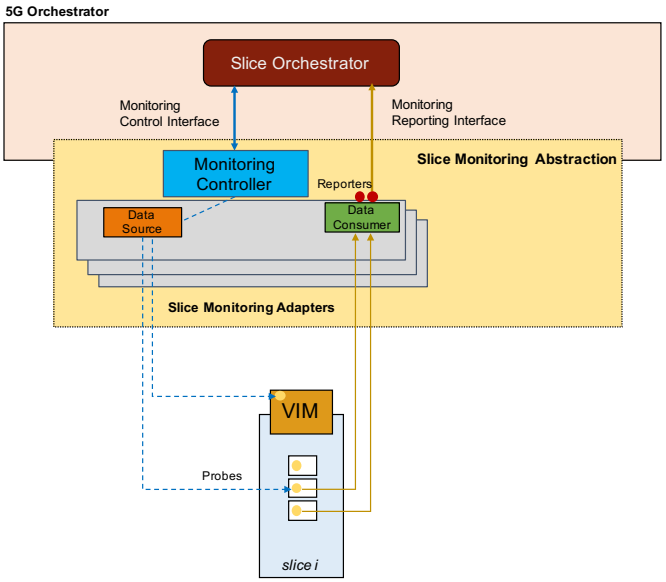

Figure 3: SMA Implementation using Lattice

managed by other orchestrators to ensure modularity, isolation, and security [8]. On the allocation of a new DC Slice, the 5G Orchestrator also allocates the required monitoring adaptation layer (in yellow) which provides a uniform measurements view of each slice regardless of both the implementation and the particular type of deployed VIM, and allows exposing the measurements for each slice.

Figure 3 highlights the way that the Slice Monitoring Abstraction (SMA) layer has been implemented using the features and components provided by the Lattice Monitoring framework. When an end-to-end slice is created by the Slice Orchestrator, different DC Slices are allocated in separate Data centers, each possibly using a different type of VIM. As the Slice Orchestrator is aware of the particular type of technology used in each DC Slice, it can interact with the SMA via its *control interface* to trigger the on-demand instantiation of a Slice Monitoring Adapter for particular that slice and VIM type.

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 implementation and / or monitoring system already available in the DC Slice and will hide the related implementation details to the Slice Orchestrator, which will be able to then consumed in a transparent way. The measurements collected from a given DC slice by the on-demand instantiated probes, will be then 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 a similar 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. Test Case: monitoring of lightweight DC Slices

In order to prove the effectiveness of the monitoring abstraction layer discussed in this paper, we have recently built a working system using the most recent version of Lattice, which was fully integrated into a Multi MANO platform discussed in [25]. This scenario is an actual deployment of the one already shown in Figure 3, where the end-to-end slices consist of the aggregation of lightweight DC slices, each running an instance of the VLSP (Very Lightweight Network & Service Platform for SDN Environments) VIM [7].

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 (see Figure 3). According to the particular topology of a slice, a new Slice Monitoring Adaptor will be instantiated on-demand for each slice segment. Then, one or more Lattice Data Source objects will be allocated in each slice (depending on the number of resources 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, in each Slice Monitoring Adaptor, the measurements collected by the probes and sent by the Lattice Data Sources will then be aggregated by a Lattice Data Consumer, which was 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 Figure 2.

Figure 4 provides an example of aggregated measurements of a VLSP DC Slice sent by a Data Consumer to the Slice Controller via the Reporting Interface.

In the Multi MANO platform, 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 (see [7] for further details). In this case, that 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 be properly structured with respect to the Lattice data model and each of the measurement object will in turn be encoded using the XDR format [22], before

```
------------------------------------------------------------
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

        id: Router-10
            cpuLoad: 76.097         cpuIdle: 590.552
            usedMemory: 507.484     freeMemory: 83.068
            energyTot: 21519.0      energyDelta: 1.1277133
            energyNow: 0.057899985

        id: Router-11
            cpuLoad: 66.075         cpuIdle: 565.153
            usedMemory: 486.419     freeMemory: 78.734
            energyTot: 20552.0      energyDelta: 1.078935
            energyNow: 0.0826233

        id: Router-14
            cpuLoad: 36.062         cpuIdle: 439.737
            usedMemory: 384.268     freeMemory: 55.469
            energyTot: 18507.0      energyDelta: 0.85627496
            energyNow: 0.077586636
------------------------------------------------------------
```

Figure 4: Aggregated measurements from a DC Slice

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.

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, 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. This is a design pattern that can be used for VIMs such as Openstack, Docker, etc., and for monitoring systems that might already be installed and up and running in the VIM or DC slice, such as Ceilometer, Prometheus, etc.

VII. Conclusions and Future works

This paper analysed a scenario linked to the 5G network landscape, which is currently gaining more attention as network providers are aiming to build flexible and dynamic Software Defined Infrastructures (SDIs) for the delivery of network services to their tenants. In this context, slicing represents a move towards an abstract segmentation of resources to achieve a more effective management via the creation of self-

contained resource bundles that can dynamically be allocated and controlled in an on-demand fashion.

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. 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 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, we fully integrated this approach into a Multi MANO (Management and Orchestration) platform. The implementation of the designed abstractions, APIs and data models was 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 work, we plan to extend the above testing scenario with the instantiation of slices whose segments include different technologies and VIMs in order to further prove the effectiveness of our proposed approach. Also directly measuring the efficiency of the data aggregation and data models would be helpful to evaluate the impact and overhead of the designed SMA layer, particularly as we use XDR in Lattice and other approaches use JSON and REST.

Moreover, future research activities on the matter of slicing will be aimed at identifying the best approach to monitor the Network Slicing parts of an end-to-end slice. This will imply reviewing and possibly extending the data models and abstractions proposed in this paper in order to achieve a unified approach for monitoring both the Data Center Slices and the different Network Slices using the same approach, and then inter-connecting them in a uniform way inside a MANO to form a view of the whole slice.

ACKNOWLEDGEMENTS

This work was partially supported by the EU projects NECOS – "Novel Enablers for Cloud Slicing" (777067).

REFERENCES

[1] 3GPP. Study on Architecture for Next Generation System, 2016.

[2] 5G Transformer. 5G Mobile Transport Platform for Verticals, 2017. http://5g-transformer.eu.

[3] 5GEx. EU H2020 - 5G Multi-Domain Exchange (5GEx) project, 2015. https://5g-ppp.eu/5GEx.

[4] 5G!Pagoda. EU-Japan - A network slice for every service, 2017. https://5g-pagoda.aalto.fi.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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 *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, June 2014.

[10] ETSI. ETSI Report on Net Slicing Support with ETSI NFV Architecture Framework. URL: http://www.etsi.org/deliver/etsi_gr/NFV-EVE/001_099/012/03.01.01_60/gr_NFV-EVE012v030101p.pdf.

[11] EU 5GPPP. White Paper on 5G Architecture, 2016. URL: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf.

[12] EU 5GPPP. White Paper on 5G Architecture, 2018. URL: https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf.

[13] IETF. Network Slicing - Revised Problem Statement draft-galis-netslices-revised-problem-statement-03, 2018.

[14] iMatix Corporation. Zero MQ Distributed Messaging, 2019 (accessed on 2019-01-07). URL: http://zeromq.org.

[15] ITU. ITU-T IMT2010/ SG13. URL: http://www.itu.int/en/ITU-T/focusgroups/imt-2020/Pages/default.aspx.

[16] ITU. ITU-T Y.3011, 2011. URL: http://www.itu.int/rec/T-REC-Y.3001-201105-I.

[17] NGMN. NGMN 5G White Paper, 2018 (accessed on 2018-10-07). URL: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf.

[18] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini. DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds. *Future Generation Computer Systems*, 29(8):2041 – 2056, 2013. Including Special sections: Advanced Cloud Monitoring Systems & The fourth IEEE International Conference on e-Science 2011 — e-Science Applications and Tools & Cluster, Grid, and Cloud Computing.

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

[20] SLICENET. End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks, 2017. https://slicenet.eu.

[21] SONATA. EU H2020 - 5G Service Programming and Orchestration for Virtualized Software Networks, 2015. https://5g-ppp.eu/sonata/.

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

[23] Stuart Clayman. Network Slicing Supported by Dynamic VIM Instantatiation. In *IETF 100, Singapore*, 2017. URL: https://datatracker.ietf.org/meeting/100/materials/slides-100-nfvrg-3-network-slicing-support-by-dynamic-vim-instantiation/.

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

[25] 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.