Design and Implementation of an Elastic Monitoring Architecture for Cloud Network Slices

André Beltrami*, Paulo D. Maciel Jr.*†, Francesco Tusa‡, Celso Cesila§,
Christian Rothenberg§, Rafael Pasquini¶ and Fábio L. Verdi*
*Federal University of São Carlos, Sorocaba, Brazil
Email: {beltrami,verdi}@ufscar.br
†Federal Institute of Paraíba, João Pessoa, Brazil
Email: paulo.maciel@ifpb.edu.br
‡University College London, London, UK
Email: francesco.tusa@ucl.ac.uk
§University of Campinas, Campinas, Brazil
{ccesila,chesteve}@dca.fee.unicamp.br
¶Federal University of Uberlândia, Uberlândia, Brazil
Email: rafael.pasquini@ufu.br

***Abstract*—A key feature of the cloud network slicing concept is the dynamic (de)provision of end-to-end infrastructures composed by computing, network, and storage resources, in order to meet the service needs of a variety of vertical industries. The resulting resource ensemble needs to be instantiated over a (potentially large) number of administrative and technological domains – a complex challenge for the management and orchestration of the allocated resources. Resource monitoring in this new so-called slice entity is extremely important in order to address the above operations. Therefore, in this work, we discuss the design and implementation of an elastic architecture for the monitoring of physical and virtual resources in cloud network slices that span both multiple administrative and technological domains. Since the concept of cloud network slices is quite recent in the literature, we highlight that our results present the performance evaluation of the proposed monitoring architecture when instantiating, stopping or performing elasticity operations on the slices. Additionally, we also analyze the performance of the architecture by scaling the number of metrics being monitored in overloaded scenarios.**

I. Introduction

The concept of *cloud network slicing* [1]–[3] is centered around the coexistence of different types of services on shared heterogeneous infrastructures to support the orthogonal demands of vertical use cases. Through emerging technologies such as 5G, SDN and NFV, combined to more consolidated ones like cloud computing, infrastructure "softwarisation" can provide the flexibility and customization required to create logical resource bundles tailored for particular use cases.

With the goal of providing end-to-end customized infrastructures to vertical customers, cloud network slices envelop, in single units, elements from different categories of resources that include cloud computing, network, and storage. The instantiation of these resources should span across multiple administrative or technological domains in an integrated manner. Once instantiated, the *tenant* requesting the creation of the slice should be able to access, configure and manage the resource bundles even with the possibility of recursively reallocating resources to other tenants or outsourcing resource management to the slice provider.

978-1-7281-4973-8/20/$31.00 © 2020 IEEE

Even though the topics of computing and networking monitoring have been widely studied in the literature, the context in which they are addressed in this paper bring valuable novelties to the community, including new challenges due to some particular characteristics of cloud network slicing embodiments. Regarding novelties and new challenges raised by this concept we can highlight: (*i*) a slice is naturally multi-tenant and multi-domain and such domains are totally heterogeneous having different VIMs (Virtual Infrastructure Managers) and WIMs (Wan Infrastructure Managers), which brings a huge heterogeneity for any monitoring solution; (*ii*) a slice is intrinsically composed by computing, networking and storage, so any monitoring solution should be capable of collecting metrics from different infrastructures; (*iii*) a slice orchestrator is capable of instantiating VIMs and WIMs on-demand during the creation of the slice and during the run-time phase by adding new resources, which is totally different and new when compared to the static provisioning of VIMs and WIMs; and (*iv*) elasticity, both vertical and horizontal, which are key features in slicing and well explored in our paper.

In this paper, we propose a monitoring architecture with the flexibility and capabilities to create, manage and stop different software components that collect measurements from the resources of a multi-domain cloud network slice. The architecture is also generic enough to support different monitoring entities. Among all the characteristics of the proposed architecture, the main one is that of being "elasticity-enabled", i.e., it is able to trigger dynamic monitoring adaptations as a consequence of a slice elasticity action. The core of the solution revolves around the use of different *Monitoring Adapters* that interact with distinct types of slice resources for timely collection of metrics.

II. Background

A. Cloud Network Slices

Cloud network slicing [1] refers to an end-to-end infrastructure composed of computing, network and storage resources. This variety of resources must be instantiated transparently to the tenant, who has the responsibility to manage and orchestrate the services that will be instantiated across the end-to-end infrastructure. Being an end-to-end concept, slices include several components that belong to different administrative domains and, therefore, demand a higher abstraction level for resource management and monitoring. In addition, the slice tenant-provider relationships introduce new business models. Next, we describe the key actors and relevant terms:

- *Infrastructure Provider*: owns and manages physical or virtualized resources offered to compose a slice required by a single or multiple tenants;
- *Slice Provider*: although it may not own a physical infrastructure, it is responsible for providing end-to-end slice services between different administrative domains, such as slice creation, decommission, orchestration, management and monitoring;
- *Tenant*: hires resources, manages and orchestrates cloud network slices and offers services to customers/users;
- *Slice Part*: one slice is composed of slice parts (one or more), and each slice part normally corresponds to one Infrastructure Provider (DC or WAN);
- *Administrative Domain*: set of resources managed by a single organization, which maintains such resources under a common administration, and may or may not use different technologies for management operations.

Several important characteristics desired in a cloud network slice can be quoted: automatic life-cycle management, unattended operation, simplified resource provisioning and management, dynamic re-provisioning, high scalability and reliability, cost-effective and rapid deployment of the network or service, the possibility of new business models, among others. In order to provide these desired aspects, there is a number of activities aiming at standardizing slices being endorsed by different entities, such as 3GPP [4], ETSI [5] and NGMN [6]. However, despite all ongoing efforts, for the best of our knowledge, there has not been any particular initiative related to the specification of a dynamic architecture for monitoring cloud network slice resources. This is precisely the gap we intend to contribute to with our architectural and implementation work.

B. Vertical and Horizontal Elasticity

In the context of cloud network slices, one of the most important operations is the elasticity of resources. Moreover, the elasticity process must consider that the slice is provisioned in a multi-domain and multi-technological environment, capable of "growing" or "shrinking" the resources dynamically. Therefore, infrastructure run-time changes require an elastic architecture for the monitoring and management of resources or services, in order to detect dynamically these alterations.

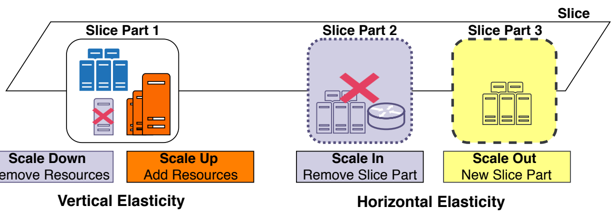

Fig. 1: Elasticity models in cloud network slicing.

In line with the resource virtualization literature [1], [7], [8], we identify two types of elasticity (vertical and horizontal) related to the way resources (computing, networking or storage) can grow or shrink in the context of cloud network slices:

- *Vertical elasticity* is the ability to dynamically resize slice parts as needed, in order to adapt for workload changes. This property expresses the ability to augment (scale up) the number of resources available inside a particular slice part (i.e., physical machines, switches, routers), when the demand for the services supported by the slice increases. On the other hand, the vertical elasticity scale down occurs when idle resources are identified and thus removed from a particular slice part.
- *Horizontal elasticity* is the ability to create (scale out) or remove (scale in) slice parts dynamically, using resources from other providers, following the need to adapt to workload changes. As the service workload increases and available slice part resources are not enough to cope with the demand, it may be possible to scale out resources by requesting the creation of new slice parts to additional infrastructure provider and then connecting them appropriately.

Figure 1 illustrates the two previously defined types of elasticity, i.e., vertical and horizontal. The former can be seen on the left side of the figure, in which slice part 1 resources are removed/added simultaneously (in purple and orange, respectively). In contrast, the representation of horizontal elasticity can be seen in slice parts 2 and 3. Slice part 2 has been removed as a result of a horizontal elasticity scale in operation (in purple) and slice part 3 has been added as a result of a horizontal elasticity scale out operation (in yellow).

III. Related Work

Despite the fact that topics related to the monitoring and management of cloud and network infrastructures are widely studied in the literature, we could not find any work in the context of cloud network slices and all novelties raised by this new entity. Therefore, the papers presented in this section focus on proposals that are somehow related to ours, as they involve resource monitoring and management or even different slice concepts.

The survey in [9] presents an overview of the term monitoring with an emphasis on cloud environments and data centers. The objective of this work is to qualitatively compare some of the most used monitoring tools, considering some of the pillars of monitoring, such as scalability, portability, multi-tenant environment, and the ability to adapt according to the tenant. These characteristics among others mentioned in the article should be considered to better meet the needs of tenants and incorporated into the context of cloud network slicing.

DASMO [10] proposes an extension of the MANO architecture to support network slicing in context of the management and orchestration of virtual network functions. The paper discusses possible changes to the MANO architecture to encompass the concept of network slicing and issues regarding the monitoring aspects (e.g., scalability). The authors describe in detail the proposed components and their responsibilities. In addition, qualitative aspects of the presented proposal are discussed. However, neither a quantitative analysis nor a proof of concept implementation of the architecture is presented.

The work presented in [11] proposes a framework for cloud monitoring. The idea is to provide a self-configured monitoring system for cloud computing, in particular running OpenStack and different monitoring tools, for example: native cloud monitoring solutions (i.e., OpenStack Ceilometer) and non-cloud monitoring solutions (i.e., Nagios, and MRTG). This framework is able to automatically create monitoring slices corresponding to the cloud infrastructure allocated in OpenStack environment. However, the slicing concept in this work is restricted to single clouds, leaving out of scope key aspects such as multi-domain and heterogeneity of resources (computing, network, storage).

The slice monitoring approach presented in [12] provides an architecture for the monitoring of end-to-end slices and a related software implementation that is specifically focused on the DC slice parts. Whilst the paper discusses the need of ensuring slice monitoring abstractions and the dynamic instantiation of monitoring adapters, it does not address the research problems arising from slice elasticity operations.

IV. Elastic Monitoring Architecture

We propose an *elastic monitoring architecture* capable of adding or removing monitoring components on-demand, as the number of resources in a slice increases or decreases. This is realized through the deployment of different slice monitoring components across multiple administrative and technological domains, transparently to the tenant. The proposed architecture aims to provide an easy way to create, stop, update and manage monitoring components for cloud network slices. The architecture is generic enough to be capable of supporting multiple monitoring entities, via technology-specific adapters that are deployed in the different administrative domains where the slice parts were instantiated.

Figure 2 presents the elastic monitoring architecture and illustrates the interaction among its components. The figure showcases two slices (one Health care slice and one Mobile Broadband slice) being monitored. Coloured in green in Fig. 2 are the static components of the architecture: Engine Controller (EC), Database (DB), and the Distribution Mechanism (DM). These components are always present and operate

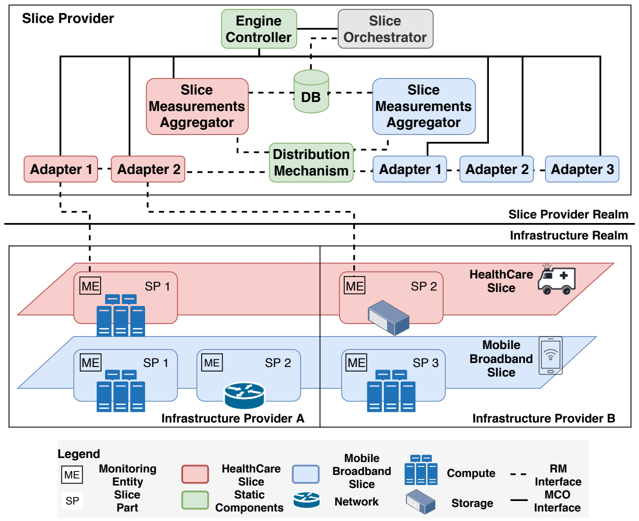

Fig. 2: Elastic monitoring architecture for cloud network slicing.

over multiple slices. The dynamic components – coloured in blue and in red for each slice – are the Slice Measurements Aggregator (SMA) and the Adapters. A set of these components' instances is (un)deployed and updated on-demand (dynamically) per slice.

The Slice Orchestrator (shown in gray in the picture) acts as the interface between our system and the Tenant, and triggers monitoring operations through the Engine Controller. For the sake of simplification, this component is out of the scope of this work and will not be detailed hereafter. The Engine Controller takes care of the monitoring process for the whole infrastructure and reacts by adequately adapting the monitoring subsystems on-the-fly, in accordance to the elasticity operations on the cloud network slices, and in a transparent way to the Tenant.

The **Engine Controller** is responsible for receiving requests from the Slice Orchestrator and performing the corresponding actions, such as deployment, decommissioning, update or re-configuration of the monitoring components. The EC offers four basic functionalities that are stated below and better described in the next section.

- ***start monitoring*** – it deploys monitoring components dynamically (Slice Measurements Aggregator and Adapters) in order to monitor a specific end-to-end slice;
- ***stop monitoring*** – it removes monitoring components dynamically from a slice that has been decommissioned;
- ***update monitoring*** – it instantiates or removes Adapters as elasticity operations are performed by the Slice Orchestrator;
- ***re-configure monitoring*** – it re-configures different Adapters at run time, e.g., dynamic probe rate adaptation, additional KPIs to be collected, temporarily switching off the collection of measurements from a given probe.

The **Slice Measurements Aggregator** is a component created on-demand and responsible for gathering monitoring metrics related to a specific end-to-end slice from different slice parts. In order to do that, it merges the different streams of monitoring data (collected by the Adapters) and (re)encodes them following a well-defined information model. The information model defined for this work includes fields related to slice and also related to the KPI being monitored. The data fields are: KPI name, timestamp, value, and slice-related tags such as slice ID, slice part ID, resource ID, and resource type (e.g., VMs, containers, switches, routers).

The **Adapters** are instantiated on-demand to interact with different Monitoring Entities (MEs) present in each slice part. The main functionality of this component is to dynamically collect the monitoring metrics for each slice part. More specifically, the EC instantiates the Adapters on-demand to collect metrics from all entities running in the end-to-end slice. The Adapters are designed to provide a technology-specific southbound interface toward the MEs they are bound to. This interface will be used to pull metrics out of the slice part by interacting with the particular ME which is running therein. This can be seen through the dotted lines between the red Adapters (1 and 2) towards the red Monitoring Entities in each slice part. For the sake of visual clarity, in Fig. 2 we omitted the arrows from the three blue Adapters towards the blue Monitoring Entities for the Mobile Broadband Slice.

The metrics in each slice part are collected by local **Monitoring Entities** (ME), i.e., monitoring tools instantiated by different infrastructure providers. Each ME collects resource metrics related to a given slice part. As examples of these entities, we can mention well-known monitoring solutions such as Nagios, Zabbix and SNMP. Open-source tools widely used in the cloud computing environment are also good examples, as Prometheus, Netdata, Ceilometer, among others.

The **Distribution Mechanism** is responsible for receiving the metrics collected by the Adapters and distributing them to specific SMAs, corresponding to individual slices. The Distribution Mechanism can be seen as an abstracted way to represent different data distribution technologies/approaches such as push/pull, publish/subscribe, etc.

The **Database** is responsible for storing all the metrics collected from the different end-to-end slices. The Orchestrator in the Slice Provider will then use those metrics in order to trigger the orchestration operations (e.g. elasticity).

Two types of interfaces can be identified in Fig. 2 based on control-oriented versus data-oriented information flows:

- *Monitoring Control and Orchestration (MCO)*: this interface is used between the Slice Orchestrator, EC, SMAs, Adapters, and DB for deploy, remove, update and reconfigure the monitoring components.
- *Resource Monitoring (RM)*: this interface is defined between the SMA, Adapters, DM, and DB to send real-time data collected by the MEs through the Adapters, all the way to the DB by using the DM and the SMA. After processing the collected data, Slice Key Performance Indicators (KPIs) are derived.

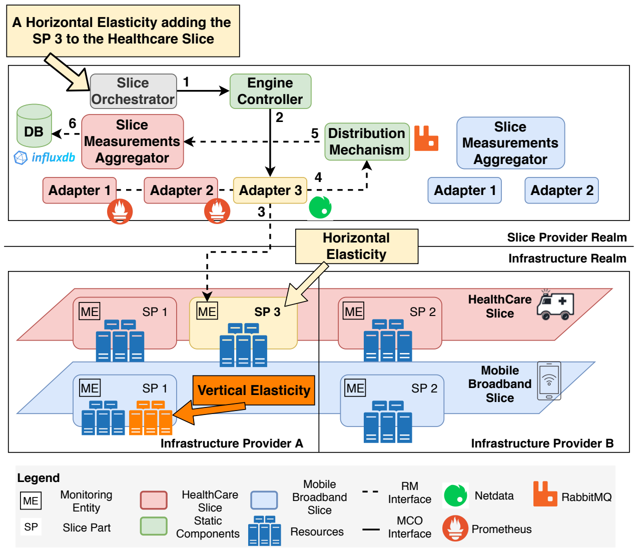

Fig. 3: Workflow of the update monitoring for elasticity events.

V. Implementation and Functional Workflows

We now turn the attention to the implementation aspects of the proposed architecture. The software modules were developed in Python 3.7, and the communication interfaces were developed as REST APIs. We use the queue system RabbitMQ to distribute the metrics from the Adapters to the SMA. The InfluxDB time series database is used to store the different metrics collected from each end-to-end slice. In order to dynamically deploy and manage all the elements of the required monitoring system, each slice has its unique set of components (SMA and Adapters), as illustrated by the red and blue components in Fig. 2.

Following a micro-services approach, SMA and Adapters are instantiated inside Docker containers and run as processes. Having a containerized implementation, we benefit from the natural features of containers such as: *(i)* every slice has its own set of isolated monitoring components as shown in Fig. 2, *(ii)* the deployment of new adapters does not affect the already running components, and *(iii)* slice elasticity (vertical and horizontal) is transparently supported by adding/removing containers (adapters).

A. Update monitoring

Upon elasticity events, the update monitoring workflows are executed. Fig. 3 shows the scale up/out processes for both vertical and horizontal elasticity actions. Note that the only type of elasticity that affects the monitoring system in place is the horizontal one. The vertical elasticity is transparent for the Adapters because, despite changing the amount of (computing, network or storage) resources, the number of slice parts stays the same. So, the ME is responsible for recognizing the new resources dynamically and to expose them back to the Adapter. This is depicted in Fig. 3 when new resources (in orange) are added to Slice Part 1 of the Mobile Broadband (MBB) Slice. In this case, the number of monitoring components in the

Slice Provider remains the same, and the associated Adapter will be able to propagate all the new metrics for that slice part transparently. However, when a new slice part is added (in yellow) following an horizontal elasticity process, a new Adapter needs to be deployed as described next (see Fig. 3).

The horizontal elasticity scale out process starts in **Step 1** when the Slice Orchestrator communicates with the EC and sends a YAML file with the characteristics of each new slice part that was instantiated. After this, the EC instantiates the required Adapters by using containers, and configures them to communicate with their corresponding MEs. As per **Step 2**, for each new slice part, a new Adapter is also instantiated. In **Step 3** of this particular example, Adapter 3 is able to communicate with the ME and collect monitoring metrics for the new Slice Part 3. It is worth noting that the SMA does not need any changes when this elasticity process occurs. **Steps 4**, **5**, and **6** share the same descriptions as steps 5, 6 and 7 from the *start monitoring process*, respectively. In **Step 4**, the new Adapters send metrics via the Distribution Mechanism. In **Steps 5** and **6**, the SMA automatically recognizes the newly produced metrics and stores them in the DB, which can be accessed by the Slice Orchestrator for resource management and orchestration tasks, including SLA re-assessment. In the case of an horizontal elasticity scale-down processes, the instantiated Adapters are removed by the EC as described in Sec. IV.

VI. Experimental Evaluation

This section describes the experiments performed to evaluate the architecture implementation. We aim at evaluating the performance of the proposed architecture when instantiating and removing monitoring components as slices are created, removed, or scaled out/in.

As such, we analyze the following aspects: (*i*) the time required to scale out/in the monitoring components when increasing the number of slice parts; (*ii*) the impact of instantiating/removing one slice by varying the number of metrics being monitored in an environment comprising multiple slices; (*iii*) the time between adding the metrics in RabbitMQ until they are inserted into the database, varying the number of slices and monitored metrics.

A. Testbed

The setup used for the evaluation is composed by two physical machines (Intel® Xeon® CPU D-1518 2.20GHz, 64 GB RAM, 2 TB HD). We instantiated a set of VMs for representing the end-to-end slices. The servers act as the infrastructure providers and have one slice part with two VMs each. The monitoring tools used in each slice part are Prometheus and Netdata, respectively. In order to represent the slice provider, we deployed the monitoring components in one HP desktop workstation, assembling a second-generation Intel i7 processor 3.40GHz, 32 GB of RAM and 1 Terabyte HD. This workstation was responsible for instantiating and managing the monitoring components (SMAs and Adapters) for all slices, as well as the Engine Controller, the Database, and the Distribution Mechanism. All tests were performed 30 times, and the metrics polling interval was set to 10 seconds.

B. Impact of the number of Slice Parts

The purpose of this graph is: (*i*) to show the average time scale in and scale out methods, by varying the number of slice parts; (*ii*) to evaluate if the adapter (Prometheus or Netdata) has a significant impact on the execution time of these methods. For this evaluation, the Prometheus and Netdata Adapters are monitoring 16 metrics per slice part.

Fig. 4 illustrates the average time to perform the scale in and scale out methods, by also varying the number of slice parts being added or deleted (1 to 100 slice parts). In other words, Fig. 4 represents the time related to instantiate/remove only the Adapters that were requested by the Slice Orchestrator in response to an elasticity operation. In the x-axis, the reader will see labels such as *sp50*, which means the creation of one slice with 50 slice parts.

We can observe that the type of adapter (Prometheus or Netdata) does not impact the execution time of the described methods, since the obtained values were very close in all graphs. Regarding the performance of the architecture, we noticed that the average time to instantiating the monitoring components in the Method Scale In with 100 slice parts (label *sp100*) was 105 seconds as can be seen in the Fig. 4, which results in ≈ 1 second per slice part.

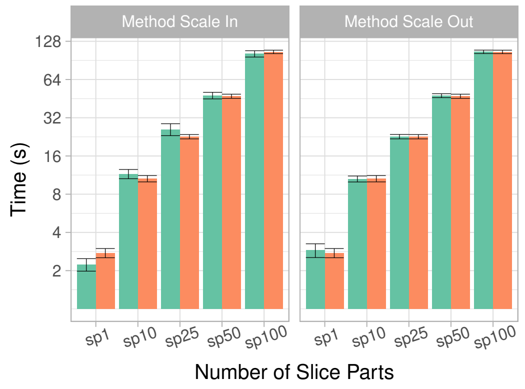

Fig. 4: Impact of the number of Slice Parts scale in and scale out methods.

C. Impact of the number of Slices

The purpose of this analysis is: (*i*) to show the impact of instantiating a new slice with just one slice part in environments that are already monitoring a set of slices; (*ii*) to analyze whether the number of metrics being monitored in new slices influences the start/stop times.

Figure 5 illustrates the execution time for the start/stop monitoring methods. In these graphs, we are evaluating the impact of starting/stopping a new slice, with just one slice part, in scenarios where the number of slices already being monitored increases according to the following values: 1, 5, 10, 25, 50, and 100 slices. Each of these already deployed

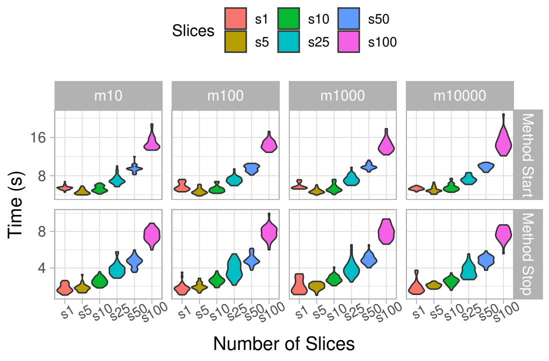

Fig. 5: Average time to run the start and stop methods in overloaded scenarios.

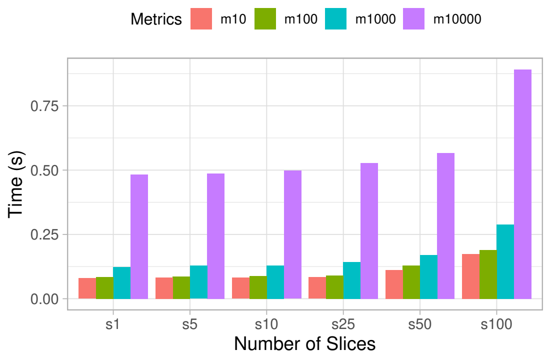

Fig. 6: Impact of number of metrics in overloaded scenarios.

slices has 2 slice parts monitored by Prometheus. In order to carry out the second analysis (*ii*), we changed the number of metrics being monitored by the new instantiated or removed slice according to the following values: 10, 100, 1000, and 10000, (labeled respectively m10, m100, m1000 and m10000).

The number of metrics being monitored in the new instantiated/removed slice did not impact the obtained results, i.e., the time to create/delete components in the proposed architecture does not change, even in scenarios where the number of metrics being monitored increases considerably. However, as the number of slices being monitored increases, the time to instantiate/remove a new slice monitoring system can also increases considerably, depending on how overloaded the environment is. For example, the time to start a single-slice-part slice in an environment with only one slice already deployed (label *s1* in red) was close to 5 seconds, regardless the number of monitored metrics. On the other hand, to create the same slice in an environment that is already monitoring 100 slices (label *s100* in pink), the average time has increased to about 15 seconds in all cases shown in Fig. 5. The stop method has a similar behavior.

D. Impact of the number of Metrics

The analysis presented in this subsection aims to show the average execution time to follow the steps 4, 5 and 6 presented in Fig. 3, Section V. These steps comprise the time required by the Adapter to add the monitored metrics (according to the proposed information model), to publish them to RabbitMQ, and finally for the SMA to consume and add them to the database. In this analysis, as the previous one, we analyze the impact of this time on overloaded environments, by varying the number of slices already being monitored in the environment, as well as the number of monitored metrics in the new instantiated slice.

Figure 6 shows the time taken to process metrics until they are inserted into the database, following the same information model described in Section IV. The time increases as the number of metrics being monitored also grows. Thus, to process and store 10000 metrics (label *m10000*) in a single-slice environment (label *s1*), it takes 0.48s on average, while the time to process and store the same amount of metrics in an 100-slice environment is 0.88s on average.

Therefore, given the results of this subsection and the previous one, we can conclude that the number of metrics does not influence the time to start or stop monitoring components of a slice, even in overloaded environments. On the other hand, the time to process metrics in overloaded environments raises as we increase the number of metrics being processed.

VII. Conclusions and Future Work

The realization of *cloud network slicing* poses many different challenges. In this work, we make a step forward on the monitoring aspects by proposing an elastic monitoring architecture that can uniformly collect metrics from the computing, networking and storage components of dynamic end-to-end slices. The main features of this architecture relate to its ability to instantiate/remove the monitoring components as slices or slice parts are created, removed and updated, also dealing with scenarios that consist of multiple administrative and technological domains.

The next steps include the evaluation of the resource utilization related to the deployed monitoring components, as well as the implementation and testing of the reconfiguration functions of both the SMA and the Adapters. We are also planning to monitor the services instantiated across the slices and integrate the proposed monitoring architecture with a real slice orchestrator developed within the NECOS project[1] capable of making intelligent decisions (e.g., resource/service allocation, slices/services elasticity, SLA violation predictions) based on the infrastructure and services metrics.

Acknowledgment

Work funded through the H2020 4th EU-BR Collaborative Call, under the grant agreement no. 777067 (NECOS - Novel Enablers for Cloud Slicing).

[1]http://www.h2020-necos.eu.

References

[1] S. Clayman, "D3.1: NECOS System Architecture and Platform Specification. V1," Tech. Rep., 10 2018. [Online]. Available: http://www.h2020-necos.eu/documents/deliverables/

[2] R. V. Rosa and C. E. Rothenberg, "The pandora of network slicing: A multicriteria analysis," *Transactions on Emerging Telecommunications Technologies*, vol. 0, no. 0, p. e3651, e3651 ett.3651. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3651

[3] P. D. Maciel Jr., F. L. Verdi, P. Valsamas, I. Sakellariou, L. Mamatas, S. Petridou, P. Papadimitriou, D. Moura, A. I. Swapna, B. Pinheiro, and S. Clayman, "A marketplace-based approach to cloud network slice composition across multiple domains," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, June 2019, pp. 480–488.

[4] X. de Foy and A. Rahman, "Network Slicing - 3GPP Use Case," Working Draft, IETF Secretariat, Internet-Draft draft-defoy-netslices-3gpp-network-slicing-02, October 2017. [Online]. Available: http://www.ietf.org/internet-drafts/draft-defoy-netslices-3gpp-network-slicing-02.txt

[5] ETSI ISG NFV, "GR NFV-IFA 022 - V3.1.1 - Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on Management and Connectivity for Multi-Site Services," ETSI ISG NFV, Tech. Rep., 2018. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[6] NGMN Alliance, "5G End-to-End Architecture Framework v2.0," NGMN Alliance, Tech. Rep., 2018. [Online]. Available: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2018/180226_NGMN_E2EArchFramework_v2.0.0.pdf

[7] A. Medeiros, A. Neto, S. Sampaio, R. Pasquini, and J. Baliosian, "End-to-end elasticity control of cloud-network slices," *Internet Technology Letters*, vol. 2, no. 4, p. e106, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.106

[8] ONF, "TR-526: Applying SDN Architecture to 5G Slicing," Tech. Rep., 2016. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Applying_SDN_Architecture_to_5G_Slicing_TR-526.pdf

[9] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, "A survey of cloud monitoring tools: Taxonomy, capabilities and objectives," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2918 – 2933, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731514001099

[10] S. Kuklinski and L. Tomaszewski, "Dasmo: A scalable approach to network slices management and orchestration," *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–6, 2018.

[11] M. B. de Carvalho, R. P. Esteves, G. da Cunha Rodrigues, L. Z. Granville, and L. M. R. Tarouco, "A cloud monitoring framework for self-configured monitoring slices based on multiple tools," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, Oct 2013, pp. 180–184.

[12] F. Tusa, S. Clayman, and A. Galis, "Dynamic Monitoring of Data Center Slices," in *5th IEEE International Conference on Network Softwarization (NetSoft 2019)*. IEEE, 2019, pp. 1–7.