Extending Slices into Data Centers: the VIM on-demand model

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

***Abstract*—This paper explores some of the mechanisms, components, and abstractions that can be utilized in order to encompass network slicing into a bigger picture for NFV delivery. In particular, we make the case for Data Center (DC) infrastructure slicing, as part of the full NFVI foundation, to ensure that the attributes prescribed to network slices are propagated into the Data Center. We show how creating a VIM (Virtual Infrastructure Manager) on-demand and dynamically allocating a new VIM for each slice, rather than having one for the whole DC, which can be beneficial for various precision scenarios.**

***Index Terms*—infrastructure slicing, VIM, network slicing**

I. Introduction

In the 5G landscape, different Infrastructure and Service Providers can explore new business models and federate their resources and service offerings in order to provide the customers with the ability of instantiating end-to-end services across multiple technological domains, which can be either located in separate geographical location of a Provider, or even outside of its own administrative borders. In order to facilitate the allocation of resources for the above end-to-end services, each 5G Provider / Operator may take advantage from using complex software systems that take care of the management, control and orchestration of all the resources to be allocated for the deployment.

The current resource orchestration model is based on a monolithic approach where resources available from different technological domains are orchestrated as a whole, and service elements allocated for different tenants usually share the same distributed physical NFVI [1] [2]. Slicing is a move towards segmentation of resources and deployment of NFV for the purpose of enhanced services and applications on a global shared infrastructure. Currently, many network slicing models are built on top of virtualization technologies [3], and have techniques that can take a slice of the network, but they do not slice the Data Center. Rather they have the NFV elements and the service elements scattered across the DC, under the control of a remote orchestrator. This placement strategy and lack of slicing is due to various reasons, but the consequence is that they do not provide a slice of the DC, in the same way as the network is sliced.

In this paper we present some of the mechanisms, components, and abstractions that can be utilized in order to encompass network slicing into a bigger picture for NFV delivery. In particular, we make the case for a new approach called Data Center (DC) infrastructure slicing, as part of the full NFVI foundation, to ensure that the attributes prescribed to network slices are propagated into the Data Center. This approach is suggested as there are some situations in which it is important to have a separate Data Center slice within a full Network Slice, including telecom scenarios with guarantees, CDN deployments, energy constraints and services with high levels privacy and isolation requirements. These ideas were first presented at the IETF 100 to the NFVRG [4].

Slices are expected to considerably transform the networking perspective and enhance software-defined architectures (e.g., SDN, NFV, etc.) by: (i) Abstracting away the lower level elements, in various ways; (ii) Isolating connectivity at a sub-network level; (iii) Separating logical network behaviours from the underlying physical network resources; (iv) Allowing dynamic management of network resources by managing resource-relevant slice configuration; (v) Simplifying and reducing the expenditure of operations; (vi) Support for rapid service provisioning; and (vii) Support for NFV deployment

The slicing approach being discussed 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 make the case for creating a VIM (Virtual Infrastructure Manager) on-demand and dynamically allocating a new VIM for each slice, rather than having one VIM for the whole DC. This strategy can be beneficial for those special situations.

To manifest this slice approach, we have designed and built a DC Slice Controller which is able to allocate a slice of a DC and create a per-slice VIM in an on-demand fashion. The DC slice and the VIM are provisioned solely for use with the service. Each slice and its associated VIM are independent of the other slices and VIMs. In this way, customers will never share servers, and the worry of having VMs of one customer interacting or spying on another customer will be eliminated. Also, the issue of one customer's VM consuming all the resources and starving other customer's VMs is also ameliorated to some extent.

The paper shows the architectural elements, designed using the aspects presented in [5], that are required to support such a model, as well as a set of layered abstractions using slicing elements, showing how they all fit together for service provisioning and integrate with an orchestrator.

II. Background

The desire to provide services on top of slices, by logically partitioning resources of a multi-domain software-defined infrastructure can be obtained through different slicing strategies depending on which system elements provide the slicing and at what layer the slicing is introduced.

Slicing at lower layers, namely the infrastructure, means that upper layers, such as VIMs and Orchestrators do not need to know about slicing. If a slice is presented to them, they can carry on working with no change or minimal change. If slicing is done in the Orchestrator, which uses an inter-domain orchestrator API interaction and /or a peer to peer approach, a slice is closer to a set of data structures in the Orchestrator rather than a partition.

Consequently, there are inherent trade-offs when selecting one or the other slicing approach. The actual decision on which slicing approach 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.

The work presented here in this paper relies on slicing at the infrastructure level. However, there are various projects and initiatives that are doing slicing at the Orchestrator level. These include SONATA [1], 5GEx [2] and 5G-Transformer [6], 5G PAGODA [7], SLICENET [8]. This approach has the easiest entry position, but is far more difficult conceptually as well as to actually implement, as all the main software elements need to be updated and adjusted to know about slices; and all of the APIs, the modules, and internal function paths, and the data structures need to be adjusted and adapted to factor in slices.

Slicing has been addressed in the past in Active / Programmable Networks research: where node operating systems and resource control frameworks produced the text Programmable Networks for IP Service Deployment [9] during the period (1995-2005). The MANA work [10] had 3 Slices Capabilities: (i) Resource allocation to virtual infrastructures or slices of virtual infrastructure; (ii) Dynamic creation and management of virtual infrastructures / slices of virtual infrastructure across diverse resources; and (iii) Dynamic mapping and deployment of a service on a virtual infrastructure / slices of virtual infrastructure.

There are various standards organisations that have been addressing slicing and creating various definitions. The ITU-T Slicing model is defined in [11], and is the basic concept of the Network Softwarization. Slicing allows logically isolated network partitions (LINP), with a slice being considered as a unit of programmable resources such as network, computation and storage. Slicing has also been addressed in ITU-T IMT2010/SG13 [12]. There is the ETSI Report on Net Slicing Support within the ETSI NFV Architecture Framework [13], the 3GPP TR23.799 Study Item on "Study on Architecture for Next Generation System" [14] which addresses Network Slicing, and the ONF Recommendation TR-526 "Applying SDN architecture to Network Slicing". The IETF Network Slicing - Revised Problem Statement [15] 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 5GPPP White Papers on 5G Architecture centred on network slicing (mark 1 - (2016)) [16]) (mark 2 - (2018)) [17].

The NGMN Slice capabilities [3] consist of 3 layers: 1) Service Instance Layer, 2) Network Slice Instance Layer, and 3) Resource layer. The Service Instance Layer represents the services (end-user service or business services) which are to be supported. Each service is represented by a Service Instance. Typically, services can be provided by the network operator or by third parties. A Network Slice Instance provides the network characteristics which are required by a Service Instance. A Network Slice Instance may also be shared across multiple Service Instances provided by the network operator. The Network Slice Instance may be composed by none, one or more Sub-network Instances, which may be shared by another Network Slice Instance.

At a testbed level, GENI [18] is a shared network testbed, i.e., multiple experimenters may be running multiple experiments at the same time. A GENI slice is: the unit of isolation for experiments; a container for resources used in an experiment; and a unit of access control. GENI experimenters add GENI resources (compute resources, network links, etc.) to slices and run experiments that use these resources. The experimenter that creates a slice can determine which project members have access to the slice (i.e., the members of the slice).

Additional characteristics, standard and research activities on Infrastructure slicing and references can be found in [19], also the 5G slicing survey [20]) have or are looking at recent approaches to slicing.

III. Data Center Slicing

In this section we discuss the concept of Data Center Slicing and present an overview of some of the concepts, mechanisms, components, that can be utilized in order to encompass network slicing into a bigger picture for NFV delivery.

We observe that slices can be requested from networks. These combined with various service elements plus NFV 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, there is an anomaly in that it is not 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. This work is predicated on the idea that slices should also be a feature that can be requested from Data Centers.

1) Data Center Slice: 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.

Moreover, a DC slice can be allocated at any Data Center: large centralised DCs, medium DCs, and mobile edge DCs. Given the ability to use a DC slice, it becomes the basis for control in virtualized environments. For most effective use in full NFVI setups and for service deployment, the following attributes of a DC slice are important:

- a DC slice needs to be as elastic as other elements in the NFVI chain, and
- a DC slice should grow or shrink dynamically at run-time under software control

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.

2) Slice Control: In order to create these DC Slices, as well as affect their elasticity, and shutdown a DC Slice, we need a control point. This is a management component deployed in each domain which can allocate a slice. We call this the *Slice Controller*. As stated, a VIM needs to be allocated for each DC Slice, and using this VIM on-demand deployment method, and its independence from other VIMs, a DC Slice owner can manage, configure, and control their own VIM.

The VIM that is deployed does not even need to be the same one that the DC owner uses for their own infrastructure. It could be one of many: e.g. OpenStack, OpenNebula, OpenVIM and can be chosen by the customer from a pre-determined catalogue of VIMs that the DC owner has. This approach is possible as a VIM is just a piece of software; we can allocate a new one any time; there is no restriction that says there can only be one VIM in a Data Center.

As the chosen VIM can be a small lightweight one, or a large one which is its own distributed system, the DC owner need to decide how and where to deploy the VIM components.

A. Structural View

In the following discussion we present figures that show the structural view of how a DC can be sliced. We see it from the viewpoint of a single Data Center. These DC slices are then composed, with network slices, into a single multi-domain topology to form a full end-to-end slice.

1) One VIM Per DC: This approach of having one VIM per Data Center, has just a single VIM for all of the resources of the DC. This is the current and common situation in Data Centers. With the one VIM per Data Center, the VMs for the NFV and other generic service elements are scattered across the infrastructure, and each slice is inter-mingled with the others. The slice customers have very little control and specification opportunities to do otherwise.

The attributes of the network part of the slice, such as isolation, reserved resources, or service guarantees, do not apply in the Data Center with the one VIM approach.

In figure 1 the different service VMs of different slices being deployed across various hosts is presented. As the picture shows, different service elements from different service instances can coexist on the same physical hosts as there is no concept of resource partitioning.

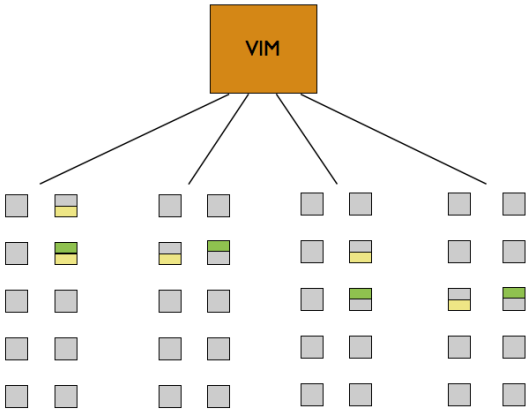

Fig. 1. Different service VMs of different slices across hosts

2) One VIM for multiple slices: The approach of one VIM for all of the resources of the DC has many issues when we introduce slices. In order to do more tasks or to do different tasks, then the VIM gets more and more functionality, more components, adapted elements, expanded APIs, and more. The VIM becomes more complex and heavyweight. Such changes would be required to add the concept of slicing to any VIM that does not support such functionality. Even if the VIM were able to deal with different slices at the VIM level, it still needs to be even more complex to deal with this.

In figure 2 one VIM for different slices is presented. However good this seems, this approach implies not only adding more complexity inside the VIM itself (as pointed out above), but it also forces all of the slices to have the same strategies and policies, as there is only one VIM for the whole DC. The owner of each slice does not have the flexibility to control the slice how they wish, for example, with different placement strategies or different approaches to energy management.

3) One VIM per Slice: In our work, we have devised an approach that overcomes these complexities, and allows us to build modular and scalable slices. This approach has many VIMs in a Data Center – one per slice. Each VIM can have its own independent strategies and can be managed differently from the other slices. The slice owner can configure their own VIM as needed for their use, for example by setting their own placement strategy [21].

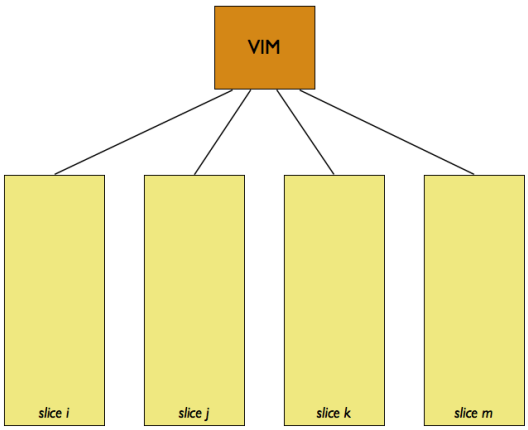

Fig. 2. One VIM for multiple slices

Having multiple VIMs running in a Data Center brings some new challenges and security issues, but these can mostly be addressed in configuration and a careful security setup. It does not require major rewrites to add slices to an existing VIM. There is a need to ensure that each slice is isolated from the others, and that a VIM of one slice cannot manage the wrong resources – namely resources in another slice.

This *one VIM per slice* is a basic premise of our approach. As a VIM is not special, and there is no requirement to have only one of them per DC, it is just another piece of software and can be started and stopped at any time. In figure 3 this setup is presented, showing each VIM having been allocated on-demand for each slice.

In the following section we present an architecture and design of a system that manifests VIM on-demand operation.

IV. Design for VIM On-demand

To facilitate the VIM on-demand approach we created a design for a system that accepts requests for slices and allocates a VIM on-demand which has been configured for the specific resource requirements, as specified by the request.

As stated, in order to create DC Slices, and update their elasticity, and shutdown a Slice, there needs to be a control point. This is a management component and we call this the *Slice Controller*. It utilizes other support components for the full functionality.

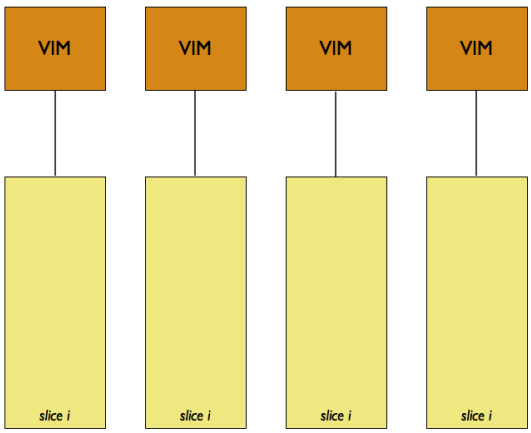

Fig. 3. One VIM per Slice, each allocated on-demand

The following components have been designed:

- a *Slice Controller* — which is the main management component that accepts requests for new slices, for slice elasticity, and slice shutdown. It is composed of a Resource Manager and a User Manager. All request commands for the Slice Controller are passed through a management interface, via a REST handler, that determines the function to be undertaken.
- a *Resource Manager* — which manages all of the resources in the Data Center that are allocated to the Slice Controller, and keeps track of which resources are free for use in a slice, which ones have been allocated to slices, and which hosts can be used for allocating new VIMs.
- a *User Manager* — that manages all of the users that can access the Slice Controller. Only validated users can request and allocate slices.
- a *Slice Information Store* — which is a database that lists all of the slices and all of the resources in the slice, together with meta-data such as the VIM REST entry point, and the keys used for access to all the resources.
- a *Slice Creator* — which is directly responsible for handling requests for slices and interacting with the Resource Manager and the User Manager to determine if it is possible to create a new slice. It asks the Resource Manager if it is possible to allocate enough hosts for the requested slice, and also if there is a host available to run the on-demand VIM, for when it is needed. If the slice creation is possible, as the resources are available, the Slice Creator interacts with the VIM Factory to instantiate the slice. If the VIM Factory succeeds then the Slice Information Store is informed to keep those details and resources for the slice, if there is a failure, the resources are released, ready for more slice requests.
- the *VIM Factory* — which is able to allocate a VIM of a particular type, and configure it to use the resources which have been picked by the Slice Creator. There is a plugin system that allows different VIM Factory objects to be used for new kinds of VIM. Each VIM Factory is responsible for creating the relevant configuration files for the specific VIM, using the list of allocated hosts, before starting the new VIM, and pointing that VIM to that new configuration. The new VIM is placed on a host under the control of the VIM Placement Manager. Once the VIM is up and running and deployed, the REST entry point can be returned to the customer.
- the *VIM Placement Manager* — which is responsible for determining which host should be used to execute a newly created VIM. Within our design, there are two main choices: (i) the VIM is allocated to run is a special space reserved for customer VIMs, or (ii) the VIM is allocated to run inside the slice that the customer has asked for. Technically there is no difference, however

there may be some differences for the customer with respect to the amount of available resource in a slice, and also the billing for the resources.

1) Run-time Setup: With a Data Center that has deployed a Slice Controller, we will observe that there are multiple VIMs each controlling their own slice. Each slice will have their own allocated hosts, each running services from different customers. No slice will be shared amongst the various customers. In figure 4, we see the different slices and the services in them, showing services in different colours.

For integration with other systems, there is a need for a high level Orchestration Platform to deal with these requests and the responses of the Slice Controller. We consider, for future operations, there to be a new component which we call a Slice Orchestrator to deal with these kinds of requests from within the Orchestration Platform. In this way, DC slices can be combined with other slices to form an end-to-end slice.

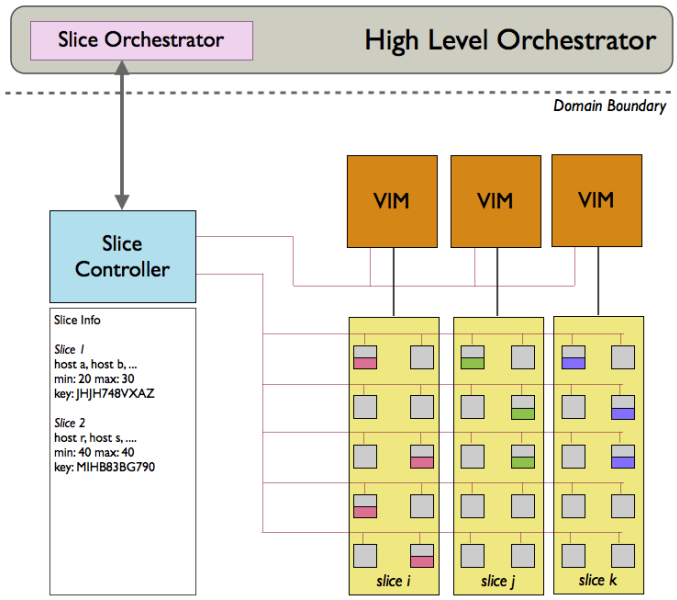

Fig. 4. Instantiation of services on different Slices

2) Allocation of new Slice: Here we show an overview of the flow for the allocation of a new slice. For the allocation a new slice, a REST call to the management interface of the Slice Controller is made. To request a new slice, the customer has to pass in attributes such as: size info, kind of low-level VIM, e.g. Openstack, Kubernetes, to the Slice Orchestrator, which will interact with the Slice Controller to actually request the slice. If successful, the resources for the slice are allocated, labelled with the Slice ID, and the Slice monitoring configured. The VIM, plus any support components, are allocated for the new slice. This could be one of OpenStack Vim + Heat or Open Mano VIM + VNFM + NFVO. A handle to the new VIM is passed back to the Slice Orchestrator, and from there the High Level Orchestration Platform binds to the newly allocated Slice and VIM. After this the Orchestration Platform can start deploying NFVs and service elements into the slice.

3) Deallocation of a Slice: The Slice Controller within the High Level Orchestration Platform is used to deal with requests and the responses to the Slice Controller in order to deallocate the slice. By deallocating a slice all of the VMs running either NFVs or service elements will be stopped, and the resources of the slice will be returned to the Data Center for use in other slices.

4) Resource Control: To ensure the smooth operation and maintain the isolation between the slices, the Slice Controller needs to inform each VIM which resources are part of the slice that is created. It uses the VIM Factory, which is responsible for creating the relevant configuration files for the specific VIM. In general, it uses a set of templates plus a list of the allocated hosts to create these configuration files. It is important that a VIM cannot manage resources that are part of another slice. This is part of the isolation principle. For slice Access Control, which has to be part of the slice management functionality, we can allocate different keys for each slice, and then use these keys to access the per-host virtualization engine.

The Slice Controller is also responsible for elasticity, and so new hosts can be allocated to a slice, or removed from a slice, at run-time under software control. The Slice Controller makes these access control updates on-the-fly as a slice can grow or shrink at runtime.

V. Results

A. Proof of Concept

To demonstrate that the concept of VIM on-demand operates as described and as expected, we devised a proof of concept implementation showing that service provisioning can be undertaken on top of a DC Slice. We have built a working implementation of the Slice Controller to support the slicing of the DC resources. To manifest this slice approach, we were able to allocate a slice of a Data Center and create a per-slice VIM in an on-demand fashion. The DC slice and the VIM were provisioned solely for use with the service, as desired. Each slice and its associated VIM are independent of the other slices and other VIMs.

For the virtualization platform we used the UCL VLSP system [22], which provides a lightweight virtual entity system that allows for the evaluation of diverse situations, including the testing of various management scenarios. The benefits of VLSP over using a full hypervisor with VMs and a full OS are: (i) better scalability, thus providing lower resource utilization and better resource allocation; and (ii) quicker startup speed and reduced heaviness, thus eliminating the issue where most of the functionality is not needed.

VLSP provides a complete environment, including its own VIM, for experimenting with scalable high-level management techniques. It has features from the service management level down to protocol stack, including a dynamic monitoring facility. VLSP allows us to experiment with a new and complete management and control facilities over virtual infrastructures using a virtual element that can be deployed as either servers and routers.

We built a Java implementation of each of the components specified in the Design section, plus the relevant supporting components. It was not a major development undertaking, as we had a clear design to work from, and the current proof of concept implementation is 26 classes and 2900 lines of code. For the VIMFactory, we built a mechanism to start the VLSP VIM which is small and lightweight, although plugins can be written for any VIM.

Some timing measures are shown, which highlight the speed of this system. For *Start Slice Controller*, this is the time from starting, for it to be ready to accept REST calls. For *Start Slice*, this is the time from the start of the REST call, doing the allocation of the slice, starting the VIM, and then returning the handle to the VIM to the caller. For *Stop Slice*, this is the time taken to stop the VIM, to release all the resources back to the Slice Controller, and respond to the caller. All operations were evaluated on a 4-core 2.6GHz Intel Core i7, with 16GB memory, running MacOS. They are:

- *Start Slice Controller* – 1.2 secs
- *Start slice* – 6.5 secs
- *Stop slice* – 5 secs

Finally, we show the REST calls used to activate the main functionality, together with the responses.

1) Create a slice: We POST in the slice request, which has the size of the slice, in this case 2 hosts, and the type of the VIM, namely VLSP.

```
curl -X POST -d '{"size":"2","type":"vlsp"}'
        http://localhost:7080/slice/
```

and here is the response, with the slice ID and the VIM details:

```
{"message":"create-slice","payload": {
    "id":1753119492, "size":2, "type":"vlsp",
    "vim": {"hostname":"washington","port":8888}
    }, "timestamp":1536257419329}
```

2) Delete a slice: We request a DELETE, passign in the slice ID, which was returned on the Slice Create call.

```
curl -X DELETE
        http://localhost:7080/slice/1753119492
```

and here is the response, with the slice ID:

```
{"message":"delete-slice","payload {
    "id":1753119492
    },"timestamp":1536257800337}
```

This is a lightweight implementation of the Slice Controller which can be used at micro DC edge cloud up to big DC. It shows that the underlying concepts work as devised, and in the following section we show how DC Slices and the Slice Controller implementation has been used in the context of large multi-domain orchestration scenarios.

B. Service Instantiation on a Multi-MANO sliced NFVI

This experiment was designed to demonstrate how network services can be deployed on end-to-end slices that include heterogeneous resources coming from different geographical locations. More specifically, the experiment rational stems in the concept of Hierarchically structured Service Provider (HSP), e.g., a scenario where a company composed of two geographically separate local offices, or divided into different business units wants to deploy network services composed of particular types of VNFs.

We suppose that the structure of the above network services is complex and their deployment needs either to leverage on resources or include specific requirements that a local office cannot offer on its own dedicated NFVI. Assuming that each local office already provides a MANO (Management and Orchestration) system to its users, the fulfilment of end-to-end services instantiation can be achieved via an inter-MANO interaction, considering a north-south interface between them, where each north-bound MANO interacts with its south-bound counterpart, as it would do with a VIM.

To further demonstrate the feasibility of the VIM on-demand approach discussed in this paper, the above mentioned HSP scenario was extended to include a sliceable NFVI on which the existing MANO systems could be utilised to perform the instantiation of network services. The orchestration functions already implemented within the utilised MANO systems did not need any modifications, whereas the adaptation components that normally interact with the whole NFVI resource layer only required minor extensions to properly trigger the on-demand creation of slices.

Figure 5 depicts a high-level view of the interactions happening in this scenario. The high level 5GEx Orchestrator operated over (1) a lightweight domain (using VLSP [22]) with slicing capabilities via a VLSP Domain Adapter and (2) a SONATA MANO through a dedicated adaptation layer, i.e., the SONATA Domain Adapter. The SONATA MANO operated on top of another partition of the NFVI, again using the slicing capabilities and on-demand instances of the VLSP VIM.

When the system was brought up, each Infrastructure Adapter initiated the preparation of the sliced resource layer in its own domain and in fact acted as the Slice Orchestrator presented in Figure 4. The Adapter drove the creation of different slices, so each one had its own on-demand instance

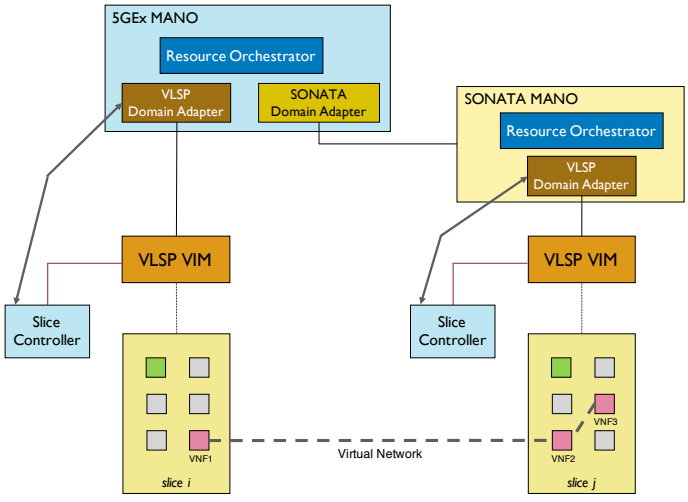

Fig. 5. Multi-MANO setup on a sliceable NVFI

of VLSP, created by interacting with the domain-local Slice Controller, and offered to the MANO's orchestration functions as an abstract interface to interact with the allocated VIM.

In particular, after a successful slice creation, each Domain Adapter received a handle to the on-demand allocated VIM and was then able to provide the required information to each Resource Orchestrator to perform the management of the service elements running on that Slice, by creating / destroying the lightweight VNFs implemented as VLSP virtual routers.

Once the different slice parts were properly set up in separate geographical locations of the NFVI, the upper layer Resource Orchestrator (i.e., the one in the 5GEx MANO) was used as the entry point for the submission of the end-to-end service represented in Figure 5. The three required service components were instantiated both in the created *slice i* (as VNF1) and *slice j* (as VNF2 and VNF3). The Resource Orchestrator was also responsible for setting up of the Virtual Network that interconnected the 3 different VNFs, to implement a Service Function Chain.

C. Independent Implementation

To further highlight the concepts laid out in this paper, the architecture and design ideas were also taken by an independent group at the Universidade Federal de Goias (UFG), Goiania, Brazil, where they built their own independent implementation of the VIM on-demand approach. In the paper [23] they reported their results.

In order to verify the time required to instantiate an infrastructure with an operational VIM, they created pre-configured virtual machine (VM) images for three distinct VIMs: VLSP [22], Kubernetes [24], and OpenStack [25]. For VLSP, they created a single VM image with all the installed components in a Debian 9 OS (size: 1.8 GB). For Kubernetes, they created two separate VM images, one for the Master Node (size: 2.1 GB) and the other for the Worker Node (size: 1.8 GB), both of these were installed in Ubuntu Server 16.04 OS's. Lastly, for OpenStack, they created a VM image for the Controller Node (size: 2.6 GB), and another VM image for the Compute Node (size: 2.6 GB), both of these were installed on CentOS 7 Minimal OS.

The UFG team then performed tests with each of the three VIMs. They collected the results over four iterations, gathering the following results:

- *Load time* – the time required to load the VM image on the nodes;
- *Boot time* – the time required to start the operating system after the image has been applied;
- *Configuration time* – the time in which the necessary settings for starting the VIM are performed;
- *Service startup time* – which represents the time for the VIM to be running after configuration.

For each VIM deployment, they performed a run of 15 separate tests, varying the number of nodes slightly in which the VIM should be deployed. The minimum number of nodes being 2, and the maximum number being 4 nodes. Figure 6 shows the mean times collected from these runs. We observe that the time to load the images (in grey) is the largest, and there is a dependency on the size of the image to be loaded, with 1.8 GB for VLSP and 2.6 GB for Openstack. However, the increase in time in relation to the number of nodes is relatively small, indicating the scalability of the solution. Boot time (in orange) is also proportional to the loaded image size, but its value is more or less the same regardless of the number of nodes used in the tests. The configuration time (in green) and the Service startup time (in blue) is dependent on the size of the codebase. We can see that lightweight VIMs such as VLSP or container managers such as Kubernetes start much quicker than the large software deployed for Openstack.

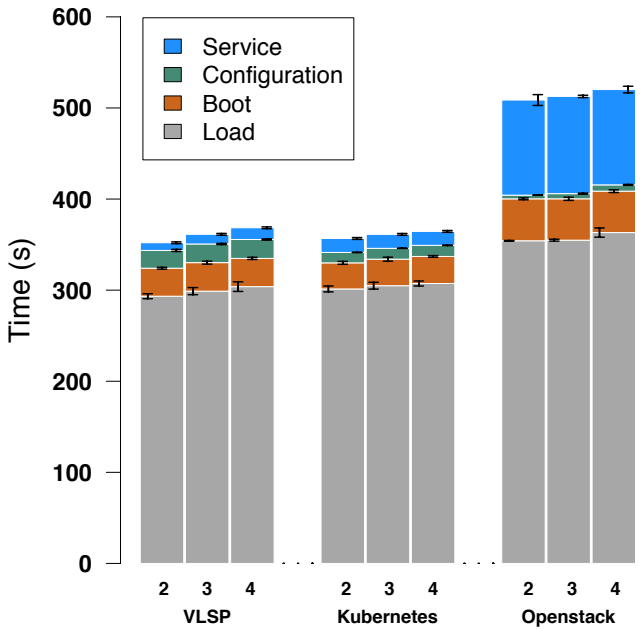

Fig. 6. Times discriminated by VIM, and number of nodes used.

This evaluation starts each of the VIM on-demand in their own virtual machine. This approach has an overhead as we see that the startup time is around 40,000 ms, which is around 6.5 minutes. Over the lifetime of a slice, this time is likely to be negligible.

VI. Conclusions

Both Data Center Slices and the VIM on-demand concept are an important contribution as there are some situations in which it is extremely important having a separate Data Center slice within a full Network Slice, such as in telecom scenarios with guarantees, CDN deployments, energy optimisation and services with high privacy and isolation requirements. Each of these slices will be allocated and de-allocated in an on-demand fashion, according to the customers' requests. A customer interacts with a Slice Controller and requests a new slice, which will result in being isolated from the other slices. Furthermore, as each slice comes with its own dedicated VIM, the customers have a new level of control and flexibility and do not have management as part of shared VIM.

In order to support service provisioning over a dynamically sliceable NFVI, it is necessary to have mechanisms to support the slicing of the network resources as well as of the Data Center (DC) compute and storage resources. Such an approach is highly suitable to the use of distributed clouds in the context of 5G networks. We see that there is a link between on-demand VIMs and NFV deployment capabilities. The VIM on-demand model does not share the hosts between slices, or allow one slice to allocate VMs to another slice. Since the slice elements are only used by the customer of the slice, this ensures isolation. These strategies are all part of a bigger picture for control and orchestration in slices.

A consequence of this DC slicing approach is that many of the DC optimization strategies which either assume access to all hosts in a DC or assume that arbitrary VMs can be packed into machines, are no longer valid. These optimizations now work at the DC slice level, not a the whole DC level. On the other hand, DC slicing allows many new opportunities for Data Center owners, including:

- the ability to deploy different VIMs for different slices, giving flexibility to the customer, and meaning the provider is not tied in to one software solution,
- the ability to request and instantiate Data Center on demand, as the customer can use an infrastructure slice as one DC, if needed.

For future work we intend to expand on the conceptual ideas founded in this paper. The desire for dynamic end-to-end slices, composed of slice parts, used for service deployment, all available at run-time under software control allows for a layered, composable abstraction for slices that is more flexible and controllable for the customer and also more flexible for the infrastructure owner. It makes it easier for the providers to allocate slices to the customer, while still giving out some level of run-time adaptability and control.

We are currently working on a full architecture for a radically different approach to the management of network slices following the layered design strategies highlighted in [5]. As we have addressed the slicing in Data Centers, we see that there is much missing in the slicing of networks. Although the network can support the infrastructure slicing operations as a native function, the management parts for dynamic, software controlled run-time operation are missing. To support the equivalent features to those presented here, we propose that networks support a WIM Slice controller, which is a software component to allocate network slices on-the-fly, and also support WIM on-demand, where a WIM will be a management entity allocated just for the network part of the slice. Using these components we will then have a symmetric architecture, which we can build new functional layers on top.

Finally, there is further work is in the area of standards, for API specification, for slice models, and for interoperability.

ACKNOWLEDGEMENTS

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

REFERENCES

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

[2] 5GEx, "EU H2020 - 5G Multi-Domain Exchange (5GEx) project," 2015, https://5g-ppp.eu/5GEx.

[3] N. Alliance, "Description of network slicing concept", ngmn 5g p1 requirements & architecture, work stream end-to-end architecture," 2016.

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

[5] S. Clayman and D. Tuncer, "Lessons from operating systems for layering and abstractions in 5g networks," in *NOMS - 2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei, Taiwan: IEEE, 2018.

[6] 5G Transformer, "5G Mobile Transport Platform for Verticals," 2017, http://5g-transformer.eu.

[7] 5G Pagoda, "EU-Japan - A network slice for every service," 2017, https://5g-pagoda.aalto.fi.

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

[9] A. Galis, S. Denazis, C. Brou, and C. Klein, *Programmable Networks for IP Service Deployment*. Artech House Books, June 2004. [Online]. Available: http://www.artechhouse.com/International/Books/Programmable-Networks-for-IP-Service-Deployment-1017.aspx

[10] A. Galis *et al.*, "Invited paper: Management and Service-aware Networking Architectures (MANA) for Future Internet," in *IEEE 2009 Fourth International Conference on Communications and Networking in China (ChinaCom09)*, Xi'an, China, 26-28 August 2009.

[11] ITU, "ITU-T Y.3011," 2011. [Online]. Available: http://www.itu.int/rec/T-REC-Y.3001-201105-I

[12] ——, "ITU-T IMT2010/ SG13." [Online]. Available: http://www.itu.int/en/ITU-T/focusgroups/imt-2020/Pages/default.aspx

[13] ETSI, "Etsi report on net slicing support with etsi nfv architecture framework." [Online]. Available: http://www.etsi.org/deliver/etsi_gr/NFV-EVE/001_099/012/03.01.01_60/gr_NFV-EVE012v030101p.pdf

[14] 3GPP, "Study on architecture for next generation system," 2016.

[15] IETF, "Network slicing - revised problem statement, draft-galis-netslices-revised- problem-statement-03," 2018.

[16] EU 5GPPP, "White paper on 5g architecture," 2016. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf

[17] ——, "White paper on 5g architecture," 2018. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf

[18] GENI, "Key geni concepts." [Online]. Available: http://groups.geni.net/geni/wiki/GENIConcepts

[19] G. A and K. Makhijani, "Infrastructure Slicing Landscape – Tutorial," in *IEEE NetSoft 2018*, Montreal, Canada, 2018. [Online]. Available: http://discovery.ucl.ac.uk/10051374/

[20] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 2429–2453.

[21] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The Dynamic Placement of Virtual Network Functions," in *1st IEEE / IFIP International Workshop on SDN Management and Orchestration*. IEEE, 2014.

[22] S. Clayman, L. Mamatas, and A. Galis, "Experimenting with control operations in software-defined infrastructures," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 390–396.

[23] L. A. Freitas, V. G. Braga, S. L. Corre^a, L. Mamatas, C. E. Rothenberg, S. Clayman, and K. V. Cardoso, "Slicing and Allocation of Transformable Resources for the Deployment of Multiple Virtualized Infrastructure Managers (VIMs)," in *Workshop on Advances in Slicing for Softwarized Infrastructures (S4SI)*, Montreal, Canada, 2018.

[24] K. Hightower, B. Burns, and J. Beda, *Kubernetes: Up and Running Dive into the Future of Infrastructure*, 1st ed. O'Reilly Media, Inc., 2017.

[25] A. Shrivastwa, S. Sarat, K. Jackson, C. Bunch, E. Sigler, and T. Campbell, *OpenStack: Building a Cloud Environment*. Packt Publishing, 2016.