Slicing and Allocation of Transformable Resources for the Deployment of Multiple Virtualized Infrastructure Managers (VIMs)

Leandro A. Freitas[*], Vinícius G. Braga[*], Sand L. Corrêa[*],
Lefteris Mamatas[‡], Christian E. Rothenberg[†], Stuart Clayman[§], Kleber V. Cardoso[*]
[*] Universidade Federal de Goias, Goiania, Brazil – Email: leandroalexandre,viniciusbraga,sand,kleber@inf.ufg.br
[†] University of Campinas, Campinas, Brazil – Email: chesteve@dca.fee.unicamp.br
[‡] University of Macedonia, Thessaloniki, Greece – Email: emamatas@uom.gr
[§] University College London, London, U.K. – Email: s.clayman@ucl.ac.uk

Abstract—**In the context of 5G networks, the concept of network slicing allows network providers to flexibly share infrastructures with mobile service providers and verticals. While this concept has been widely investigated considering mostly the network issues, in this work we focus on a slice as a service model that takes into account the data center (DC) perspective. In particular, we propose an architecture where DC slices are created over transformable (compute and storage) resources, which can be virtualized or de-virtualized on-demand. Then, on top of each slice, an on-demand VIM is instantiated to control the allocated resources. As a realization of this architecture, we introduce the DC Slice Controller, a system able to deploy and delivery full operational VIMs based on generic templates. We evaluate the effectiveness of the proposed system deploying three VIMs (VLSP, Kubernetes, and OpenStack) over commodity hardware. Experimental results show that the DC Slice Controller can timely provide a slice even when dealing with sophisticated VIMs such as OpenStack. As an example, we were able to delivery a fully functional OpenStack in four nodes in less than 10 minutes.**

I. Introduction

5G networks are expected to trigger new business models, allowing the entry into the market of vertical industries (e.g., automotive and manufacture) as well as over the top (OTT) service providers that will operate on the top of the network infrastructure based on a set of predefined service level agreements (SLAs) [1]. In this scenario, 5G networks will have to support different type of services (e.g., enhanced mobile broadband, massive machine-type communications, and ultra-reliable and low latency communications) with very distinct needs but over the same infrastructure.

Clearly, a one-size-fits-all architecture will not be able to realize this future services. In order to cope with the above requirements, the concept of network slicing has been proposed as a mean of creating logical, full-functional partitions of network infrastructures tailored for the type of service to be provided [2]. Each network slice represents a virtualized independent end-to-end network allowing network providers to deploy different architectures concurrently. Thus, the concept of slice is a natural abstraction to separate multiple tenants that operate on the same physical resources.

At the same time, driven by the need to support a flexible, scalable, and elastic mobile network, network providers are deploying cloud-like infrastructures to host Virtualized Network Functions (VNFs). For example, Telefónica has announced a plan to virtualize its network domains using a virtualized infrastructure built from commodity cloud tools and hardware sit on regional data centers [3]. This infrastructure will support IT-like applications (e.g., value-added service, operational support systems, and online charging system) as well as network-specific functions such as a virtualized core function, cloud RAN, and virtual customer premises equipment (vCPE) [4].

In this context, network slicing is one of the components of a bigger picture for delivering 5G services. Another important component to fully realize these services is the cloud or data center (DC) slicing, as it will ensure that the attributes prescribed to the network are also propagated into the data centers that host the services. While network slicing has been widely investigated in the last few year [5], [6], [7], cloud slicing and how it combines with network slicing still lacks discussion in the context of 5G communication networks.

Within the NECOS (Novel Enablers for Cloud Slicing) project, we envision a *slice as a service* (SlaaS) model to fill this gap. In our proposal, a slice is a grouping of physical and/or virtual (network, compute, storage) resources that can accommodate service components independently of other slices. The enhanced management for such a slice enabled infrastructure creates both DC and network slices on-demand, and connects and (re)configures them as appropriate to provide the end-to-end service.

Complementary to this view where optimized slices are dynamically created for a given service or category of services is the notion of on-demand Virtualized Infrastructure Managers (VIMs) automatically deployed to control and manage the resources of a given slice. Differently from traditional cloud computing setups where a single VIM is usually instantiated for the whole infrastructure, an on-demand VIM is dynamically deployed for each slice. As a consequence, the

VIM itself becomes a slice parameter and can be chosen and configured according to the service needs. For example, some services can be better satisfied by lower level virtualization tools such as Xen or KVM, while others will be better served by container platforms such as Docker or Kubernetes. Another advantage of creating on-demand VIMs is that they can be instantiated on the top of the slice, and as such, they do not need to be aware of the slice abstraction. As a consequence, vanilla versions of traditional solutions such as OpenStack, Kubertenes, OpenNebula, and OpenVIM can be used without modifications.

The architectural elements that are required to support the SlaaS and the on-demand VIM model, as well as how they all fit together for service provisioning is introduced in [8]. In this paper, we enhance that work by i) introducing the concept of *transformable resources*, that is, those which can be virtualized and de-virtualized on-demand; ii) detailing the design of one of its component, the DC Slice Controller, which is able to instantiate a DC slice and deploy an on-demand VIM on top of the slice; and iii) implementing, as a proof of concept, a prototype of this system using generic templates of three distinct VIMs (OpenStack, Kubernetes, and VLSP). We evaluated our prototype through experiments with commodity hardware in which the three VIMs were deployed. The experimental results show that the DC Slice Controller can timely provide a DC slice even when dealing with sophisticated VIMs such as OpenStack. For example, we were able to delivery a fully functional OpenStack in four nodes in less than 10 minutes. The DC Slice Controller can operate either in large clouds or less resourceful edge data centers.

The rest of this paper is organized as follows. Section II presents the related work. Section III reviews the main concepts, components, and abstractions required to support SlaaS in a multi-tenant environment. The DC Slice Controller architecture is detailed in Section IV. Section V describes the prototype and experimental results. Finally, Section VI concludes the paper and outlines future directions on this work.

II. Related Work

There are two categories of related work to be considered for our approach of slicing data center resources in the context of 5G network systems. The first category is about network slicing, while the second involves testbed based solutions focused on dynamic provisioning supported by image management and deployment.

In the first category of related work, the Third Generation Partnership Project (3GPP) has defined a new network architecture for network slicing support [9]. This architecture is service-based, meaning that whenever possible, the architectural elements of the system are defined as network functions that offer their services via interfaces to other ones that are allowed to consume their services. In this architecture, a network slice refers to a set of features and functionalities that form a complete network service for the user equipment. Slices are instantiated from predefined templates that define specific functions to be instantiated for a given service requirement. Tenants request a slice based on the templates available in the catalogue. An arbitration entity then, grants or denies the requests based on resource availability. Once a slice request is granted, its relevant information is propagated to the appropriate network components and the slice is installed. Several works have been proposed based on the 3GPP 5G system architecture. A centralized capacity broker was introduced in [10] in charge of network slicing admission control operations. This component was significantly improved in [11] to support optimal allocation and configuration of Radio Access Network (RAN) slices based on on-demand network slice requests. In [12], the authors presented a system architecture in charge of creating network slices in massive IoT scenarios building on IoT Brokers features and a 5G Network Slice Broker. However, the 3GPP system architecture specifications define network slicing only within the scope of 3GPP specified resources, i.e., that what specifically composes a public land mobile network (PLMN) [2]. Thus, data center resources are not explicitly considered in the architecture.

Some notable approaches directly focused on network architectures supporting network slicing and VNFs are research projects such as 5G NORMA [13] and SONATA [14]. The 5G NORMA project introduces a 5G system architecture based on the concepts of network slicing, network function decomposition, and software-defined mobile network control. In this architecture, a logical network is decomposed into individual functional blocks. A network slice then becomes a composition of some of the functional blocks linked into a chain. Two Software Defined Network (SDN) controllers are then designed to control the full set of functional blocks, one for the functional blocks that are common to multiple slices and the other for those blocks that are dedicated per slice. The SONATA project focuses on providing a service programming and orchestration system to develop, deploy, and orchestrate VNFs. The concept of network slicing is also supported through a slice manager plugin that can either use an external slice orchestration system or implement slice management functionalities by itself directly controlling SDN forwarding elements in the network. In both projects, however, network slicing is concerned to network resources and do not address data center resources explicitly.

Still in the context of networking slicing, it is important to mention some works that consider data center resources as part of the core network [15], [16]. However, in such works, the data center function is limited to host VNFs. The mechanism and abstractions to support the slice concept inside the data center is not discussed. Other works [17], [18] analyze the resource optimization problem related to networking slices and thus are as a complement to our work.

In the second category of related work are a number of testbed management solutions that allow the provisioning of software stacks based on image management tools. This includes Emulab [19], FIBRE [20], and FutureGrid [21]. In these solutions, one or more level of abstractions allow the automatic deployment of software stacks on a large number of virtualized and non-virtualized resources. This is usually achieved by creating template images that are stored in a common repository and adapted according to the environment in which the image will be deployed. Similar to these works, our DC Slice Controller uses generic templates to instantiate on-demand, fully operational VIMs. However, our work target data center environments instead of testbeds. Additionally, the DC Slice Controller provides the customization of the template according to the information provided by the tenant before delivering the VIM.

III. Providing Slices as a Service

Clayman [8] introduces an overview of the mechanisms, components, and abstractions that can be utilized in order to provide a SlaaS model based on dynamic VIM instantiation. In this view, a network provider owns and operates the physical infrastructure, which includes computation, communication and storage resources. Verticals and OTTs, also known as tenants, pay for providing their services over the network provider's infrastructure, which is shared among these entities. Since each tenant has his own service requirements, a subset of the resources is made available for him as a *slice*, and each slice is isolated from the others.

In Clayman's model, a slice can be seen as a collection of physical resources spread around many data centers, including large centralized, medium, and mobile edge data centers. These resource are then dynamically connected at run-time to create an end-to-end networked system. Within a data center, the set of resources allocated to a slice is called a *DC slice* and usually encompasses compute and storage resources. In addition, a DC slice has all the properties expected from a slice abstraction, that is, i) it can be controlled and managed independently from any other DC slice; and ii) it can grow or shrink dynamically. However, different from other approaches designed for cloud federation, in Clayman's model, each DC slice has its own VIM that is deployed on-demand on top of the DC slice. As a consequence, the VIM choice is not a feature pre-determined once by the network provider, but can be an option for the tenant. It also means that the tenant now can manage, configure, and control their own VIMs.

Clayman's model also define a similar slice abstraction for network resources, referred to as *Network slice*. A network slice is created on-demand to connect two DC slices. Thus, it can be seen as a set of links that connects two DC slices. As part of the isolation principal, a network slice is controlled and managed independently from any other network slice. It is also managed and configured by its own Network Infrastructure Manager (NIM), which is deployed on-demand once the network slice is instantiated.

In summary, in Clayman's model a slice is a collection of DC slices connected by Network slices, each one with its own VIMs and NIMs. This view is illustrated in Figure 1 together with the main layers needed to support it. In the following, we first describe these layers, then we highlight the steps required to create a slice using the functional elements of these layers.

- Orchestration Layer: manages the slice lifecycle (i.e. slice instantiation, maintenance, and termination) and coordinates the different DC and Network slices of a given slice to act as a single abstraction to ensure an optimized allocation of the necessary resources and connectivity.
- Slice Control Layer: acts as a point of control and management of DC and Network slices. For each data center, there will be a DC Slice Controller in charge of creating DC slices within the data center, allocating the required compute and storage resources, and deploying an on-demand VIM based on the tenant specification. Similarly, for each network domain, there will be a Network Slice Controller in charge of instantiating a network slice between two DC slices and deploying the on-demand NIM.
- Infrastructure Layer: comprises all physical resources required to delivery 5G services including cloud nodes, networking nodes together with their associated links.

Using for instance a portal, the tenant can request a slice with certain properties and service level details. This request is sent to the Orchestrator that maps the slice specification into compute, storage and network resources and decides on which parts of the infrastructure to allocate them. The Orchestrator then contacts all the relevant infrastructure Slice Controllers (DC and Network) and ask them to create a slice in their domains. Each Slice Controller allocates the necessary resources, instantiates the on-demand VIM/NIM to manage the resources of the allocated slice part, and returns to the Orchestrator the details required to access the VIM/NIM. The Orchestrator interacts with the VIMs and NIMs to glue the separate slice parts together, completing the slice topology. Finally the service is deployed over the slice. During its lifecycle, the slice can grow or shrink to accommodate the demand. When the slice is terminated, the resources are deallocated.

Transformable Resources

In Clayman's model, a DC slice comprises bare metal resources and thus, no tenants share physical resources. Although the demand for bare-metal cloud services has increased given preferences for services with high levels of performance and security [22], limiting the resource allocation to this single model may be too restrictive.

To overcome this limitation, in this work, we propose the concept of *transformable resources*, that is, a resource that can be used on isolation (bare metal) or shared (virtual), depending on the demand. Examples of bare metal resources include servers, switch or routers, while virtual resources (VR) are virtualized instances of a bare metal hardware. These instances are created through hypervisors such as XEN, KVM, FlowVisor, OpenVirtex. Figure 2 illustrates this concept.

A transformable resource can thus be virtualized or devirtualized on-demand. To implement this concept, the hypervisor itself has to be automatically installed/unstalled on-demand. Again, the use of generic templates and images can help on this task.

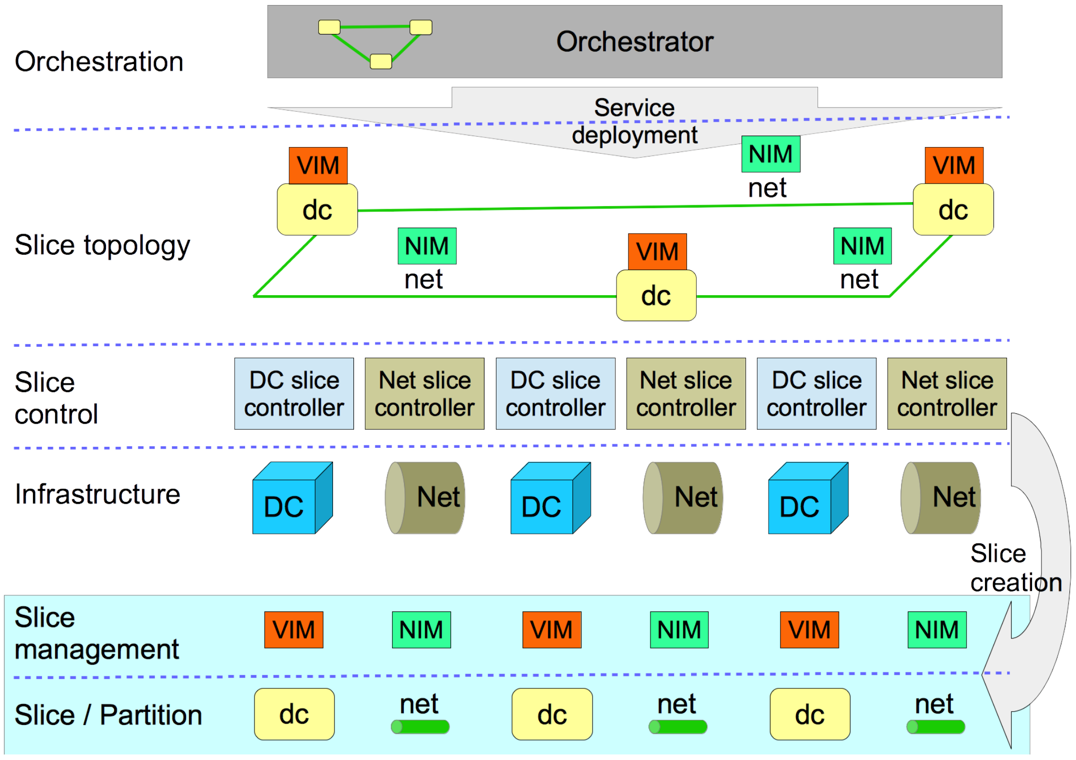


Fig. 1. Big picture for delivering slices as a service.

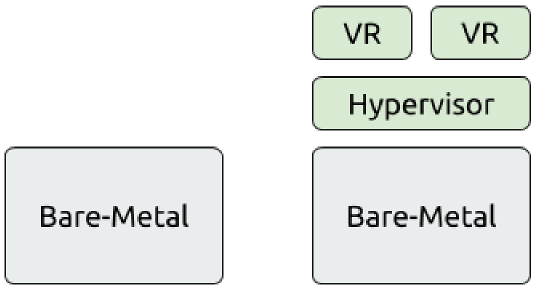


Fig. 2. A resource that can acts as bare metal or virtual depending on the demand.

Applying the concept of transformable resources to Clayman's model, three types of DC slices are possible:

- Type I, formed by bare metal resources only;
- Type II, composed by virtual resources only; and
- Type III, a combination of bare metal and virtual resources.

Figure 3 shows two examples of DC slices Type I on the left, and two examples of DC slices Type III on the right. We can see that the DC slices 1 and 2 do not share physical resources, while DC slices 3 and 4 do. Naturally, DC slices Type II and Type III are VIM dependent, i.e, they can only instantiate VIMs that support creating virtual resources inside a virtual resource (e.g. creating containers inside virtual machines). Example of such VIMs are Kubernetes and VLSP.

Interestingly, scenarios for DC slices Type II and Type III are becoming more frequent. The obvious use case is development environments. For example, the development environment of a network provider could be instantiated as a DC slice Type II. Similarly, production-ready environments can also benefit by using such slice types as lightweight virtualization technologies are as a complement to traditional ones.

IV. DC Slice Controller

This section describes our design for a DC Slice Controller system. Particularly, in this work, we focus on creating DC Slices Type I. DC Slice Types II and III will be addressed in future work.

Figure 4 illustrates the general architecture of the DC Slice Controller. In the figure, we distinguish three layers: the Slice Controller Interface, the slicing service layer, and the resource (infrastructure) layer.

The Slice Controller Interface allows administrators (network providers) and users (tenants) to interact with the system. Using this interface, the administrator can maintain the data center resource inventory while tenants can define DC slice requests.

At the center of the architecture is the slicing service layer, which actually creates the DC slices and instantiates on-demand VIMs based on the tenant request. The first component of this layer is the Slice Manager, composed by two subcomponents: User Manager and Resource Manager. The Resource Manager manages all of the resources in the data center and keeps the resources and their states up to date.

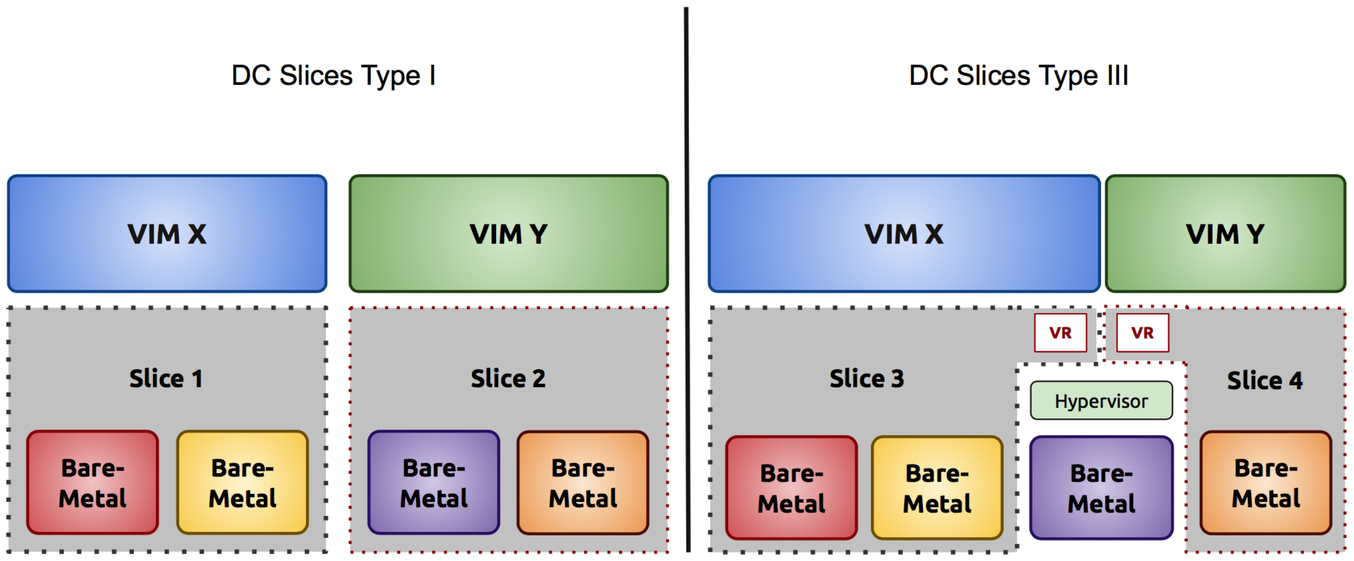


Fig. 3. Examples of DC slices Type I and Type III.

The User Manager component manages all of the users that can access the DC Slice Controller in that site, confirms a user's identity, verifies their permissions and prevents undue inappropriate access to these resources.

The Slice Creator receives slice requests from the Slice Controller Interface and interacts with the Resource Manager and the User Manager to determine if it is possible to create a new slice. To this end, The Resource Manager interacts with the Slice Information Store, which keeps track of which resources have been allocated to which slice. If the slice creation is possible, the Slice Creator registers the new slice (and resources) in the Slice Information Store and contacts the VIM Factory.

The VIM Factory component 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. In order to actually deploy the appropriate VIM template, the VIM Factory interacts with the Placement Management component, which is in charge of choosing which of the selected nodes will run specific VIM components (e.g. master and worker components). The Placement Management decides the role of each selected node and return the decision to the VIM Factory. Then, the VIM Factory deploy the VIM component templates in the appropriate nodes and configures them.

V. Prototype Implementation and Experimental Results

We implemented The DC Slice Controller system using a set of technologies. We developed The Slice Controller Interface component using a web-based user interface built on top of Bootstrap [23], along with AngularJS [24]. The implementation of the internal components of the slicing service layer has been implemented using the Control and Management Framework (OMF) version 6, specially the Resource Management Framework (or OMF Broker) and the Resource Controllers (or RCs) [25], [26]. The OMF framework contains the required tools to load pre-configured images, allowing the installation, configuration and execution of software stacks in a timely manner. In the following, we detail the web-based user interface (Slice Controller Interface), the testbed we have setup to conduct the experiments and results.

The web interface performs REST calls to the OMF Broker in order to perform the operations related to the slice creation. Figure 5 illustrates the web interface for creating a slice. Figure 5(a) illustrates the interface where the tenant enters the most basic information related to the slice, such as slice name, slice owner, data and time period in which the slice must exist, and a description of it. After this, the tenant selects the type of resources that will compose the slice and their configurations in term of processing and memory capacity. Figure 5(b) and Figure 5(c) show the selection of the amount of resources related to bare metal and Virtual Machine, respectively.

After selecting these resources, the tenant chooses the VIM he wants for managing the slice resources. Figure 6 presents a situation in which Kubernetes is selected and some customized information is provide in order to properly configure the template. For each type of VIM, there will be a set of information that is specific to the tool. The IETF [27] describes a technology independent information model for network slicing. As next steps, we will include new parameters beyond those presented in Figura 5. This will allow the network operator to define in greater detail the requirements of slice users.

In order to validate our implementation of the DC Slice Controller system, we setup a testbed composed of one server hosting the DC Slice Controller components and four other nodes. This testbed is illustrated in Figure 7. All nodes have the same hardware configuration, which allows images

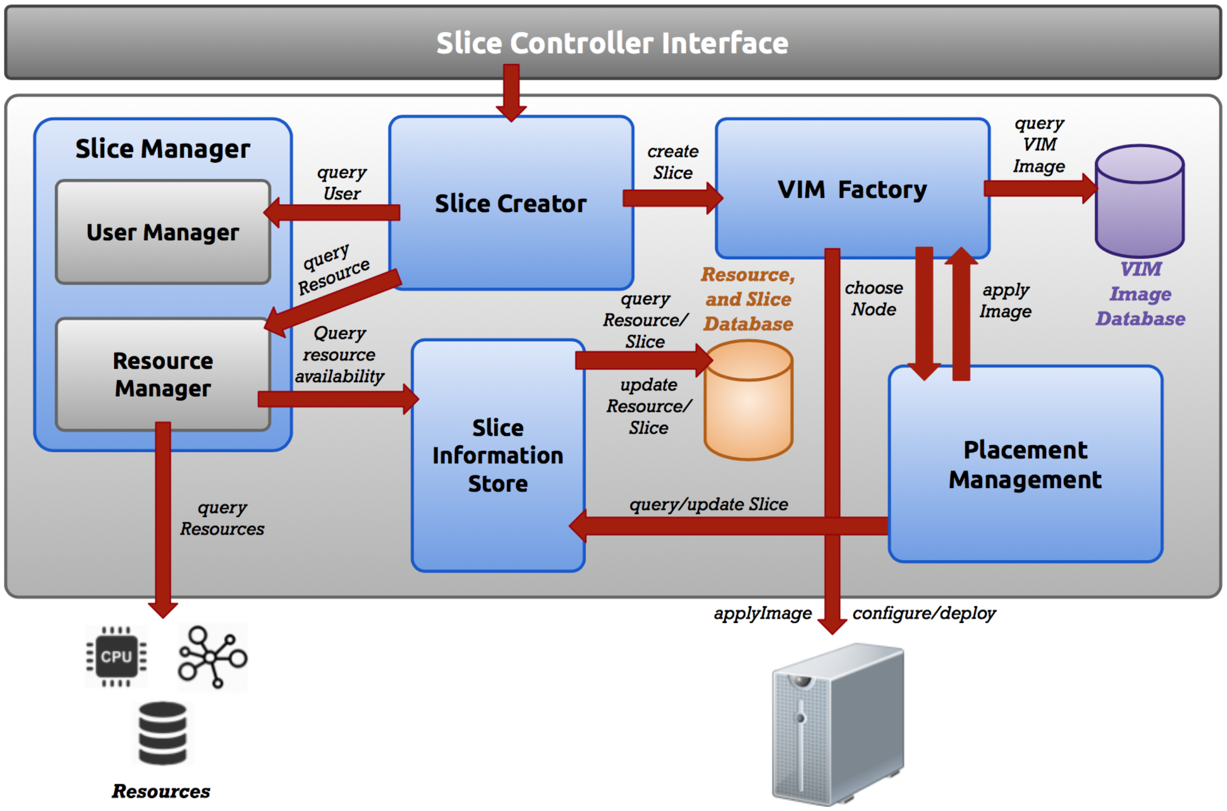


Fig. 4. DC Slice Controller Architecture

generated on one node to be compatible with the others. Each node has an Arduino, called Chassi Management Controller (CMC), that is connected to the CMC network. The CMC is used to turn on and turn off the node, as well as to check the status (on/off) of the node. The other interface of the node is connected to the Control network, from which the VIM images are installed and from which the node communicates to the other nodes.

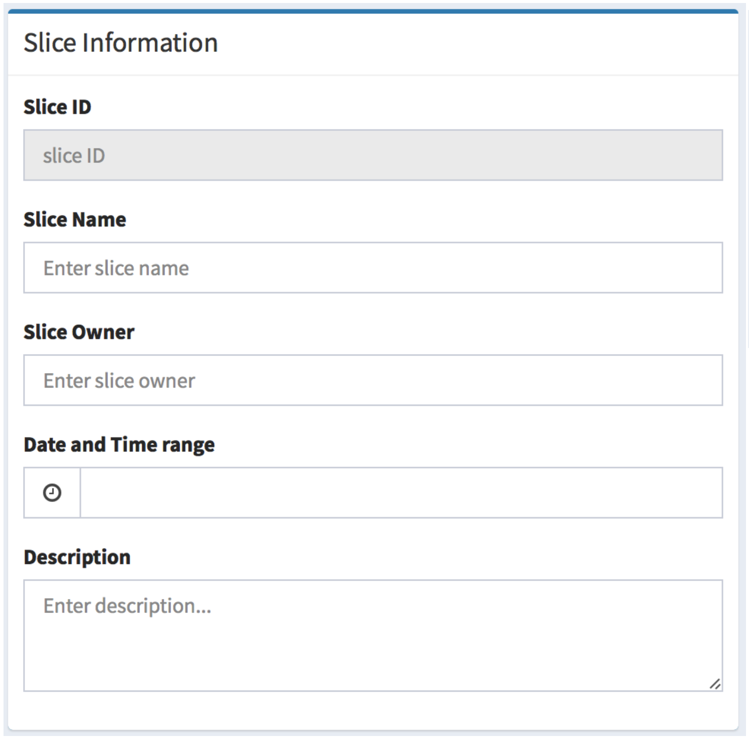
We created pre-configured images for three distinct VIMs: VLSP, Kubernetes, and OpenStack. For the VLSP, we created a single image with all the installed components in Debian 9 (1.8 GB). For the Kubernetes, we created two separate images, one for the Master Node (2.1 GB) and the other for the Worker Node (1.8 GB), both installed on Ubuntu Server 16.04. Lastly, for the OpenStack, we created an image for the Controller Node (2.6 GB), and another image for the Compute Node (2.6 GB), both installed on CentOS 7 Minimal.

In order to verify the time required to instantiate an infrastructure with an operational VIM, we perform tests with each of the mentioned VIMs. We collect four times: *i)* Load time, the time required to load the VIM image on the nodes; *ii)* Boot time, the time required to start the operating system after the image has been applied; *iii)* Configuration time, in which the necessary settings for starting the VIM are performed; and *iv)* Service startup time, which represents the time for the VIM to be running after configuration.

For each VIM, we performed 15 tests varying the number of nodes in which the VIM should be deployed, the minimum being with 2 nodes and the maximum being with 4 nodes. Figure 8 shows the mean times, with the 95% confidence interval, collected in our tests. As illustrated, the time to load the images is the largest, and it depends on the size of the loaded image. However, we can observe that the increase in time in relation to the increase in the number of nodes is relatively small, indicating the scalability of the solution. Boot time is also related to the loaded image, and its value is virtually the same regardless of the number of nodes used in the tests.

The process of configuring nodes with VLSP and Kubernetes considers the following steps on each of the nodes: configure the network and restart the network service. In our tests, the time to restart the network service was higher in Debian 9 than in Ubuntu Server 16.04, and therefore the setup time of the VLSP service was higher than that of Kubernetes. The OpenStack configuration is more complex, and involves the following steps: *i)* to configure static IP, and hostname of the nodes; *ii)* to change the configuration files with new IPs; *iii)* to update the Compute Nodes list, and; *iv)* to change the admin password of OpenStack.

The time taken for OpenStack to be operational from its initialization is greater than the time of the other VIMs. This happens because it is composed of several different modules


Slice Information

Slice ID

slice ID

Slice Name

Enter slice name

Slice Owner

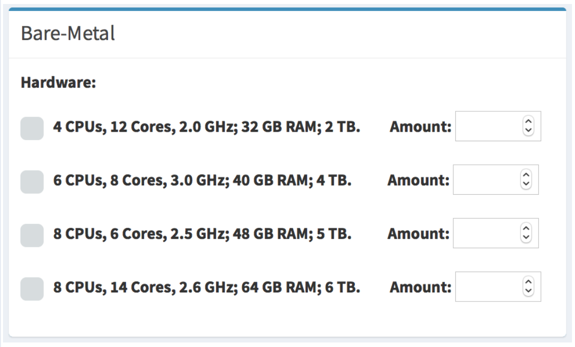
Enter slice owner

Date and Time range

Description

Enter description...

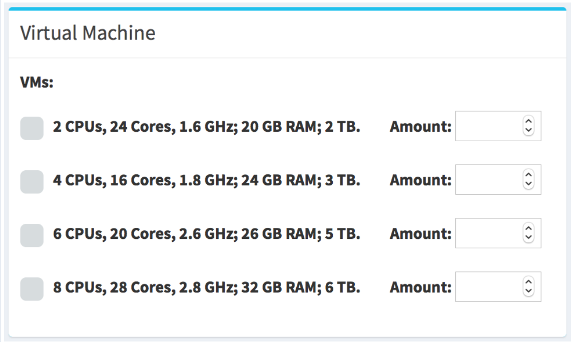
(a) Slice Information


Bare-Metal

Hardware:

4 CPUs, 12 Cores, 2.0 GHz; 32 GB RAM; 2 TB.	Amount:
6 CPUs, 8 Cores, 3.0 GHz; 40 GB RAM; 4 TB.	Amount:
8 CPUs, 6 Cores, 2.5 GHz; 48 GB RAM; 5 TB.	Amount:
8 CPUs, 14 Cores, 2.6 GHz; 64 GB RAM; 6 TB.	Amount:

(b) Selecting Hardware

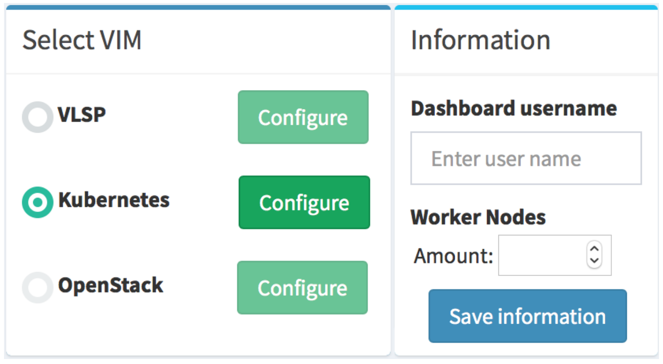

Virtual Machine

VMs:

2 CPUs, 24 Cores, 1.6 GHz; 20 GB RAM; 2 TB.	Amount:
4 CPUs, 16 Cores, 1.8 GHz; 24 GB RAM; 3 TB.	Amount:
6 CPUs, 20 Cores, 2.6 GHz; 26 GB RAM; 5 TB.	Amount:
8 CPUs, 28 Cores, 2.8 GHz; 32 GB RAM; 6 TB.	Amount:

(c) Selecting VM

Fig. 5. Interface for creating a slice.


Select VIM

VLSP [✓ Kubernetes] OpenStack

Information

Dashboard username

Enter user name

Worker Nodes

Amount:

Fig. 6. Interface for selecting a VIM

(e,g., Compute Service, Keystone Service, SQLDB, RabbitMQ and Neutron Service), which makes the full service take longer to start. VLSP is the simplest VIM, simulating many of the features of a real VIM, so it presents the smallest service startup time.

Finally, we ran tests loading two different VIMs at the same time to show the ability of the DC Slice Controller system to split the infrastructure into slices. Each of the VIMs was loaded onto two nodes. In this sense, we perform the process of deployment these images using multicasting. To do this, the nodes start, enter PXE mode (Preboot eXecution Environment), and the Slice Controller initiates the process of loading images. This operation is performed from a Frisbee [28] server process that is running on the Slice Controller. Thus, nodes execute client processes that make requests to the Slice Controller to load the images.

In Figure 9, we show the load, boot, configuration, and service startup times for each set of tests and discriminated by VIM. As we can see, the times to install the VIMs together are very similar to those required to install each VIM separately. This shows that creating two slices simultaneously does not affect the system performance significantly.

VI. Conclusion

In this paper, we introduced the concept of transformable resources and described how this concept wides previous ideas on cloud slicing. We implemented and evaluated a system for slicing resources in a DC. Our system is also able to deploy and customize traditional VIMs, e.g., OpenStack or Kubernetes, in their vanilla versions. Our experimental results have shown that our system is able to quickly deploy customized VIMs in different slices.

As future work, we plain to continue our implementation efforts to complete the entire DC Slice Architecture following our design. We also plan to evaluate the deployment of traditional VIMs over a combination of physical and virtual resources. We also want to investigate resource allocation problems that may raise in a DC composed of transformable resources under time evolving demands.

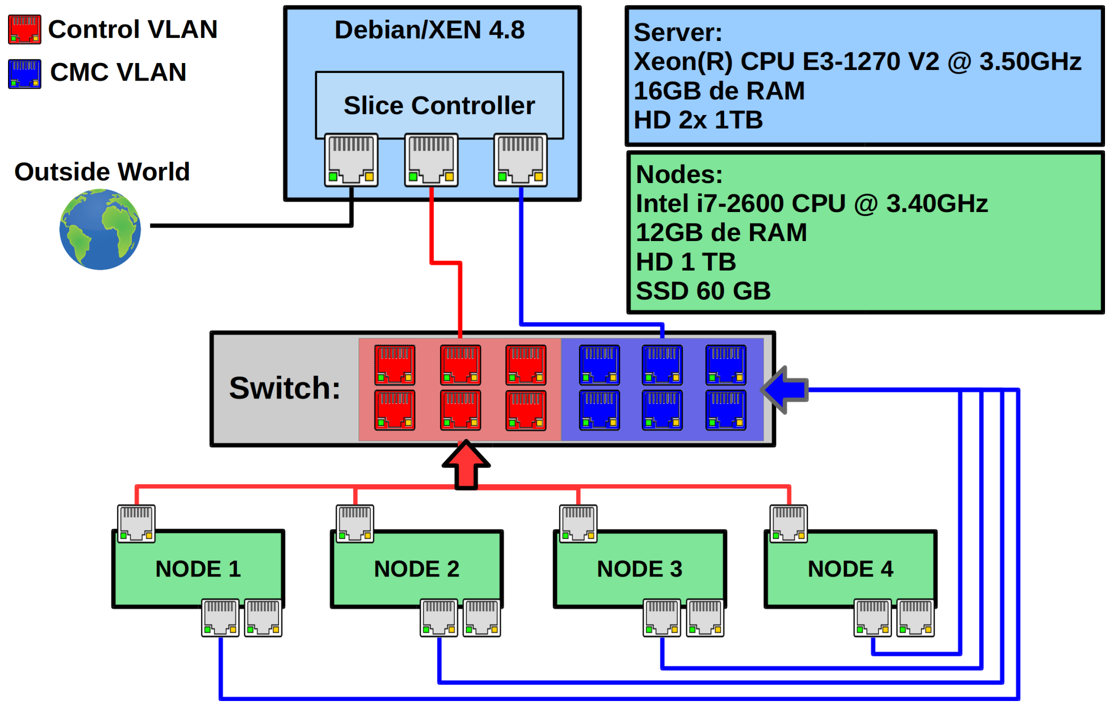


Fig. 7. Testbed used in our experiments.

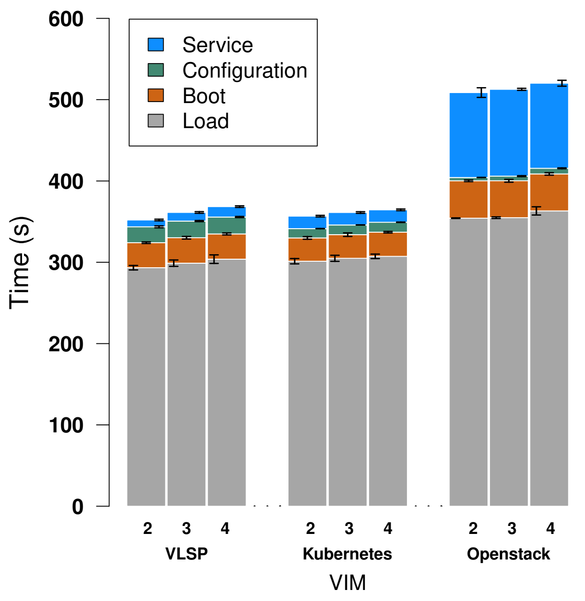


Fig. 8. Times discriminated by VIM, and number of nodes used.

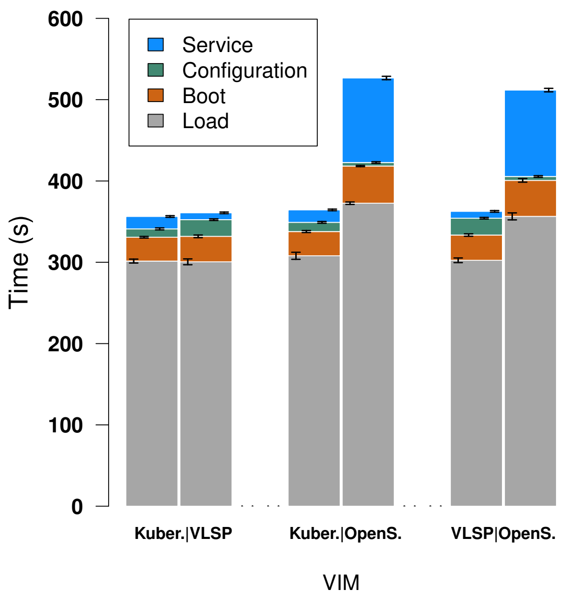


Fig. 9. Times discriminated by set of VIMs used in the tests.

ACKNOWLEDGEMENTS

This research was partially supported by the H2020 EU-Brazil joint call grant agreement no. 777067 (NECOS - Novel Enablers for Cloud Slicing) funded by the European Commission and the Brazilian Ministry of Science, Technology, Innovation, and Communication (MCTIC) through RNP and CTIC. The authors would also thank João P. Esper, Murillo Nunes and Elton Vivot for their contributions generating the VIM images and performing the experiments.

REFERENCES

[1] 5G PPP Architecture Working Group. (2017) View on 5G Architecture (Version 2.0). [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2017/07/5G-PPP-5G-Architecture-White-Paper-2-Summer-2017_For-Public-Consultation.pdf

[2] Third Generation Partnership Project (3GPP). (2017) System architecture milestone of 5G Phase 1 is achieved. [Online]. Available: http://www.3gpp.org/news-events/3gpp-news/1930-sys_architecture

[3] Analysys Manson. (2017) Telefónicaś UNICA architecture strategy for network virtualization. [Online]. Available: http://www.analysysmason.com/telefonica-UNICA-architecture-strategy-for-network-virtualisation-report

[4] Luis M. Contreras and Diego R. López, "A Network Service Provider Perspective on Network Slicing," *IEEE Softwarization*, 2018.

[5] Ravindran, Ravishankar and Chakraborti, Asit and Amin, Syed Obaid and Azgin, Aytac and Wang, Guoqiang, "5G-ICN: Delivering ICN Services over 5G Using Network Slicing," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 101–107, 2017.

[6] Katsalis, Kostas and Nikaein, Navid and Schiller, Eryk and Ksentini, Adlen and Braun, Torsten, "Network slices toward 5G communications: Slicing the LTE network," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 146–154, 2017.

[7] Nikaein, Navid and Schiller, Eryk and Favraud, Romain and Katsalis, Kostas and Stavropoulos, Donatos and Alyafawi, Islam and Zhao, Zhongliang and Braun, Torsten and Korakis, Thanasis, "Network Store: Exploring slicing in future 5g networks," in *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*. ACM, 2015, pp. 8–13.

[8] Stuart Clayman. (2017) Network Slicing Supported by Dynamic VIM Instantatiation. [Online]. Available: https://datatracker.ietf.org/meeting/100/materials/slides-100-nfvrg-3-network-slicing-support-by-dynamic-vim-instantiation/

[9] Third Generation Partnership Project (3GPP). (2017) System Architecture for the 5G System. [Online]. Available: http://www.3gpp.org/ftp//Specs/archive/23_series/23.501/

[10] Samdanis, Konstantinos and Costa-Perez, Xavier and Sciancalepore, Vincenzo, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[11] V. Sciancalepore and K. Samdanis and X. Costa-Perez and D. Bega and M. Gramaglia and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[12] V. Sciancalepore and F. Cirillo and X. Costa-Perez, "Slice as a Service (SlaaS) Optimal IoT Slice Resources Orchestration," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–7.

[13] P. Rost and C. Mannweiler and D. S. Michalopoulos and C. Sartori and V. Sciancalepore and N. Sastry and O. Holland and S. Tayade and B. Han and D. Bega and D. Aziz and H. Bakker, "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, 2017.

[14] S. Draxler and H. Karl and M. Peuster and H. R. Kouchaksaraei and M. Bredel and J. Lessmann and T. Soenen and W. Tavernier and S. Mendel-Brin and G. Xilouris, "SONATA: Service programming and orchestration for virtualized software networks," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2017, pp. 973–978.

[15] P. Rost and A. Banchs and I. Berberana and M. Breitbach and M. Doll and H. Droste and C. Mannweiler and M. A. Puente and K. Samdanis and B. Sayadi, "Mobile network architecture evolution toward 5G," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 84–91, 2016.

[16] Nikaein, Navid and Schiller, Eryk and Favraud, Romain and Katsalis, Kostas and Stavropoulos, Donatos and Alyafawi, Islam and Zhao, Zhongliang and Braun, Torsten and Korakis, Thanasis, "Network Store: Exploring Slicing in Future 5G Networks," in *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, ser. MobiArch '15, 2015, pp. 8–13.

[17] S. Vassilaras and L. Gkatzikis and N. Liakopoulos and I. N. Stiakogiannakis and M. Qi and L. Shi and L. Liu and M. Debbah and G. S. Paschos, "The algorithmic aspects of network slicing," pp. 112–119, 2017.

[18] Z. Xu and W. Liang and A. Galis and Y. Ma, "Throughput maximization and resource optimization in NFV-enabled networks," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[19] Emulab. (2018) Emulab - Network Emulation Testbed Home. [Online]. Available: https://www.emulab.net/

[20] Rede Nacional de Ensino e Pesquisa - RNP. (2018) Future Internet Brazilian Environment for Experimentation. [Online]. Available: https://fibre.org.br/

[21] von Laszewski, Gregor and Lee, Hyungro and Diaz, Javier and Wang, Fugang and Tanaka, Koji and Karavinkoppa, Shubhada and Fox, Geoffrey C and Furlani, Tom, "Design of a Dynamic Provisioning System for a Federated Cloud and Bare-metal Environment," in *Proc. Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit*. Citeseer, 2012.

[22] Im, Jaeseong and Kim, Jongyul and Kim, Jonguk and Jin, Seongwook and Maeng, Seungryoul, "On-demand Virtualization for Live Migration in Bare Metal Cloud," in *Proceedings of the 2017 Symposium on Cloud Computing*, ser. SoCC '17. New York, NY, USA: ACM, 2017, pp. 378–389.

[23] Bootstrap. (2018) Bootstrap Documentation. [Online]. Available: https://getbootstrap.com/docs/4.0/getting-started/introduction/

[24] AngularJS. (2018) AngularJS API Docs. [Online]. Available: https://docs.angularjs.org/api

[25] Rakotoarivelo, Thierry and Ott, Maximilian and Jourjon, Guillaume and Seskar, Ivan, "OMF: a control and management framework for networking testbeds," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, pp. 54–59, 2010.

[26] Stavropoulos, Donatos and Dadoukis, Aris and Rakotoarivelo, Thierry and Ott, Max and Korakis, Thanasis and Tassiulas, Leandros, "Design, architecture and implementation of a resource discovery, reservation and provisioning framework for testbeds," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2015 13th International Symposium on*. IEEE, 2015, pp. 48–53.

[27] IETF, "Technology Independent Information Model for Network Slicing," Tech. Rep., 2017. [Online]. Available: https://tools.ietf.org/html/draft-qiang-coms-netslicing-information-model-01

[28] M. Hibler, L. Stoller, J. Lepreau, R. Ricci, and C. Barb, "Fast, scalable disk imaging with frisbee." in *USENIX Annual Technical Conference, General Track*, 2003, pp. 283–296.