On Edge Cloud Architecture and Joint Physical Virtual Resource Orchestration for SDN/NFV

Shah Nawaz Khan and Roberto Riggio

Wireless & Networked Systems (WiN)
FBK CREATE-NET
Via Alla Cascata 56/D, Trento 38123 Italy
{s.khan, r.riggio}@fbk.eu

Abstract. 5G networks will incorporate virtualized network infrastructures and new technologies such as Software Defined Networking and Network Function Virtualization. The focus of these developments has predominantly been in the core and back-haul network segments. However, the ambitious Key Performance Indicators (KPIs) set for 5G networks will necessitate a renewed focus on the network edge in view of the virtualized infrastructure and SDN/NFV adoption. Several challenges are anticipated to be addressed for cloud at the edge, prime among which is the heterogeneity that spans, among others, hardware, software, radio, networking and virtualization domains. In this paper, we investigate cloud at the edge architecture, its unique challenges and the technology enablers. We present a prototype heterogeneous edge cloud implementation with focus on joint physical and virtual resource management and orchestration while supporting multi-tenancy for edge services. A scheduling & orchestration module is presented that interfaces with Kubernetes cloud management system to support service deployment per two unique polices for load balancing and energy saving. We present details of the considered edge cloud platform, the scheduling & orchestration module and its functions inside the edge cloud. Finally, we present some preliminary results and comparisons of the implemented orchestration policies in the context of heterogeneous edge services.

Keywords: Edge Cloud, SDN/NFV, Orchestration, Kubernetes

1 Introduction

Wireless communication is an integral part of modern day lives across the world. In the past decade, requirements on wireless networks to support increasing number of new services beyond the simple voice calls and connecting larger number of end-users have been consistently intensifying. Until recently, mobile communication networks have been addressing these requirements and challenges with continuous evolution of the radio access technologies and by incrementally increasing the overall network capacity. However, with the recent focus on and proliferation of social media, networking, augmented and virtual reality and Internet of Things applications, the requirements and challenges have intensified

such that mere evolution of 4G networks will not fully address them. These requirements and challenges are more stringent and diverse ranging from ultra low latency, to massive machine type communications to enhanced multimedia and broadband applications. Moreover, the pace at which the new network services and applications are developing, it is difficult for network operators to keep pace with by upgrading the core infrastructure, back-haul and radio access capacities. It is anticipated that in the next few years, the annual global IP traffic volume will reach an unprecedented 3.3 ZettaBytes (1 ZB = 1000 Exabytes) [1]. These numbers and the associated network requirements have made 5G, the next big step in the mobile wireless communication networks, one of the most important areas of research and development. Several key technologies are anticipated to play a major role in defining the 5G networks' characteristics including Cloud based virtualized network infrastructure, Software Defined Networking (SDN) and Network Function Virtualization (NFV). These technologies and their practical realization has been an area of immense interest among researchers from both academia and industry. A large number of initiatives have been taken under the 5G networks R&D umbrella developing virtualization, SDN/NFV, Management and Network Orchestration (MANO) solutions [6], [5] [7]. Most of these initiatives have targeted the core and back-haul network segments with centralized cloud infrastructures. The network edge, that is, the radio, networking and communication infrastructure closest to the end-users (Access Points, Base Stations) has not received the attention it warrants. Some of this can be attributed to the issues such as complexity, lack of standardized platforms, node heterogeneity, distributed nature of the edge infrastructure and the large number of Points of Presence (PoPs). These factor make it challenging to develop and manage cloud infrastructure for the edge and orchestrate the resources for multiple tenants as anticipated for 5G networks. Providing cloud architectural blueprints for resource constrained edge devices, technology components for virtualization and resource orchestration (both physical and virtual) will help bring the benefits of virtualization and SDN/NFV closer to the network edge. In this paper, we focus on lightweight virtualization and resource management at the network edge. We characterize the unique features of network edge, the limitations of state of the art solutions and present a prototype heterogeneous edge cloud with a joint physical and virtual resource orchestrator module. The orchestrator can pro-actively scale the physical and virtual resources available to multi-tenant services per a given policy of load-balancing or energy saving. Our implementation is based on Kubernetes [2] container orchestration system and integrates with its main control elements for placing VNFs at the edge. The prototype edge cloud consists of nodes having different hardware architectures and computing resources integrated into a unified cloud fabric supporting service deployment, multi-tenancy, resource isolation and scaling. We present details of the hardware platform, the cloud virtualization stack and the orchestrator module and evaluate its performance. The rest of the paper is organized as follows. Section 2 discusses state of the art on cloud virtualization and SDN/NFV related technologies in view of their limitations for the network edge. Section 3 details the main contribution

of this paper and presents an edge cloud architecture and virtualization stack addressing key constraints and function requirements. Preliminary results from the evaluation platform are presented in section 4. The paper concludes with a summary and an outlook for future work in section 5.

2 State of the Art

Recently, the network edge has attracted significant interest from researchers as evident from the abundant literature available under synonymous concepts but different terminologies. Mobile Edge Computing, FoG Computing, Multi-Access Edge Computing, Cloudlets etc., are a few well-known umbrella terms which in essence, aim to realize computing and network services closer to the end-users. In spite of these parallel developments in nomenclature and predominantly theoretic research work, there is a general dearth of clarity on the platform for edge computing such as hardware, software and architecture. A common consensus from the SDN/NFV community is that the benefits of cloud and virtualization must be realized at the network edge to support 5G use-cases and features such as multi-tenancy, resource slicing and orchestration etc. However, the available virtualization and resource orchestration technologies tailored towards data-center oriented centralized clouds and Virtual Machine (VM) based VNF abstraction do not suit the constraints of the network edge. For creating the cloud abstraction, the existing commercial and open source hypervisor technologies for SDN/NFV such as VMware ESX and ESXi, Microsoft Hyper-V, Xen, KVM etc., have a large hardware resource requirements footprint [8]. Usually in order of multiple high power processing cores and Gigabytes of memory, these hypervisors are virtually impossible to deploy on dispersed and resource constrained nodes. From the 5G networks and SDN/NFV perspective, the additional management and control elements required in the ETSI MANO architecture [4] such as Virtual Infrastructure Managers (VIM), VNF Manager (VNFM) and NFV Orchestrators need to be deployed. Many realizations of these components such as OpenDaylight, OSM, OpenO, OpenBaton etc., are tied to the centralized data-center oriented cloud model. The hardware resources (storage, compute, networking) required for realizing the same cloud architecture at the network edge are literally unavailable as edge nodes are resource constrained devices and are spread out over larger geographical areas. The level of hardware abstraction realized for centralized clouds where VNFs are completely devoid of information about the underlying hardware features may also prove limiting for the edge. For certain services, it may make more sense to expose the features available at the nodes such as hardware architecture, computing resources, dedicated hardware etc., to tailor their scheduling towards those nodes. Another limiting factor for cloud at the edge using existing virtualization solutions is the VM based VNF realization. A VM adds many layers of software abstraction (e.g., guest OS, unnecessary libraries) between the physical processor and the actual software that carries out a virtual network function. This makes VM based VNF realization unsuited to the limited resources available at the edge cloud. Moreover, with

large sizes of VM based VNFs and the limited memory and storage resources of individual edge nodes, the scheduling and VNF placement problem becomes extremely complicated. Another constraint that proves a hindrance is the dispersed nature of edge nodes and the variance of networking links among them (throughput, firewalls etc.). This necessitates a more distributed realization of edge cloud rather than the centralized solutions available for core SDN/NFV implementations. A federation umbrella could tie together dispersed edge clouds and provide an interface to service providers through which network services could be deployed across distributed edge cloud. Finally, a number of advances have been made in the cloud resource management and cloud-native software design domains which have not been considered in the cloud for SDN/NFV context. These developments include the micro-services based software design, DevOps, Serverless Functions etc., which place a higher emphasis on resource utilization efficiency, robust cloud-native designs and rapid deployment. These developments are more suited to the edge cloud context and call for a design where VNFs are composed of small services interfaced together to create the end-to-end network service.

3 Edge Cloud Architecture and Resource Orchestration

Considering the challenging KPIs for 5G networks addressed in section 1 and the limitations of tailoring the existing virtualization solutions to the network edge addressed in section 2, it can be argued that cloud at the edge must be based on lightweight virtualization technologies and must integrate heterogeneous hardware while supporting the 5G service requirements. Furthermore, the ETSI MANO stack where specific control elements such as VIM, VNFM and NFVO are responsible specific SDN/NFV management tasks should be realized, potentially in a converged form and with small resource requirements footprints. Considering these and the recent developments in the cloud and virtualization domains such as Micro-Services Architectures, DevOps and Function as a Service (FaaS) etc., where resources are utilized to their fullest, we condense the requirements of edge clouds into following list.

- Heterogeneity: Cloud at the edge must integrate hardware nodes of different architecture and resources (compute, storage, memory, networking) into a unified abstraction providing similar interfaces to the service providers as in centralized cloud infrastructures. Moreover, cloud at the edge should hide the complexity of these underlying hardware differences from the deployed services i.e., a service provider should not be forced to design the service template according to the hardware architecture of the host infrastructure. Any mapping between the requirements of a VNF and the capabilities of the underlying hardware should be done by the cloud control and management functions.
- Resource Awareness: Cloud at the edge should be several order of magnitude lighter in terms of hardware requirements compared with centralized cloud technologies. This implies that the edge cloud and MANO features should be

realized using virtualization technologies that can run efficiently on hardware resource constrained devices deployed in end-user premises.
- MANO Features: MANO is an essential requirement from the SDN/NFV management and control point of view. Service providers must have a simple interface through which they can deploy an edge resident network service and manage it in isolation from other services. Therefore, the essential features of VIM, VNFM, and NFVO must be realized albeit considering the discussed resource constraints and heterogeneity of the network edge. Moreover, management polices for cloud resources (physical and virtual) and tenant-specific service orchestration must be supported.
- Services: Cloud at the edge must support the distinctive features of centralized SDN/NFV clouds such as resource slicing, multi-tenancy, service isolation, auto-scaling etc. Moreover, support for new cloud-native software design principles such as micro-services architecture should be supported to utilize the dispersed computing resources more effectively.

Considering these requirements and constraints, we focus on container based virtualization where VNFs can be realized in container images instead of VMs. Containers have a significantly low overhead not only in terms of storage, memory and processing requirements compared with VMs but they also eliminate the need for the hypervisor layer. This resource efficiency usually comes at the cost of security and service isolation but several measures can be taken to address these concerns including container specific Linux kernels and more sophisticated access policies. Beyond the container based VNF abstraction, we use Kubernetes to emulate the role of VIM and NFVM by extending it to support (a) heterogeneous hardware nodes, (b) custom scheduling and orchestration module and (c) allowing for mixed service deployment in architecture agnostic manner. The remainder of this section details the considered virtualization stack and MANO feature realization.

3.1 Physical Architecture & Virtualization Stack

Figure 1 depicts a subset of the physical architecture of edge cloud infrastructure and the virtualization stack used in this work. The hardware platform comprises five ARM architecture based Raspberry Pi 3 nodes and two Intel x64 based Laptops assembled in a cluster and local network. All nodes run platform specific Linux distribution and a container runtime daemon (Docker in our case). The selection of Docker in general and container based virtualization in particular was made considering the resource limitations and hardware heterogeneity of the nodes. To integrate the nodes into a cluster where services can be deployed at cloud abstraction level, we use Kubernetes [2]. Kubernetes is an open source container management and orchestration platform that can run on several types of physical nodes and has a considerably small resource requirements footprint compared with hypervisor and VM based virtualization platforms. The core Kubernetes control elements such as Etcd, Controller Manager, API Server, DNS and default Scheduler run in Docker containers on a single Raspberry Pi node.

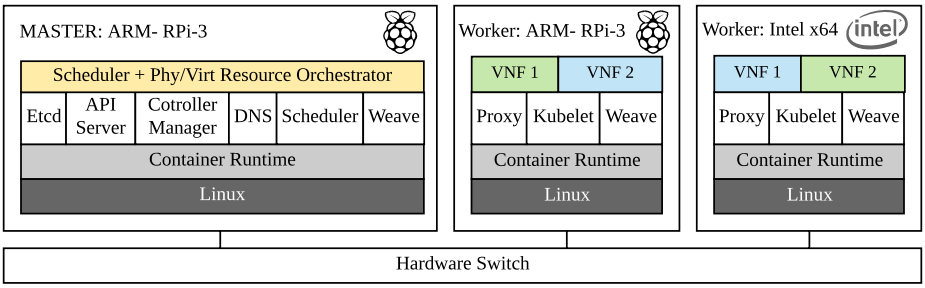

Fig. 1. Physical architecture and virtualization stack of the edge cloud

The weave element is deployed alongside Kubernetes core control elements to provide overlay connectivity among service VNFs. The details of these control elements is beyond the scope of this work and can be found in [2]. Kubernetes uses a Master-Worker based architecture where the master node hosts all the main control plane elements while the worker nodes host a subset which enable communication with the master node. We host the master node functions on a Raspberry Pi and integrate the rest of ARM and x64 nodes as workers. However, by default, Kubernetes does not integrate nodes of different hardware architecture into a single cloud abstraction i.e., the worker nodes must be of the same architecture as the master node. This limitation arises during the Kubernetes installation where the master node downloads control plane elements container images on the worker nodes according to its own architecture. Therefore, if the master node elements are instantiated on an ARM node as in our case, Kubernetes will download ARM based images for the rest of cluster members including on the x64 nodes. To overcome this problem, we modified the configuration for master node allowing it to download architecture specific images on worker nodes. To this end, we modified the control daemon configuration for kube-proxy by duplicating it for x64 architecture and specifying the correct architecture images.. This problem is expected to be natively addressed in future Kubernetes versions where control plane images are expected to be architecture specific using the Docker image manifests.

Most of the ETSI MANO control functions can be realized inside the Kubernetes cluster including VIM, VNFM and NFVO by either using the default Kubernetes control elements or by extending the cluster with new control elements. We utilize both approaches in our prototype edge cloud setup. For the VIM and NFVO functions, we have developed a dedicated scheduling and orchestration module (detailed in the next subsection). This module, depicted in yellow color in figure 1, takes the role of VIM by elastically scaling the infrastructure resources to the requirements of the deployed services following a scheduling policy. To this end, the available physical resources of the nodes and requirements of the deployed services are actively monitored. The same module is responsible for network service orchestration including deployment, scheduling VNFs, chaining and scaling. The VNFM role is shared between the Kubernetes

control elements that actively probe the state of deployed VNFs and restart a service VNF if detected to be dysfunctional. Finally, to ensure service isolation and that tenants are allocated a specific set of resources, both physical and virtual, we use the Kubernetes namespaces feature together with labeling of the nodes. A Kubernetes namespace is a virtual container for deployed services in the edge cloud which can enforce resource restrictions on the services falling in a particular namespace. With namespaces and node labels, a service can not only be confined to specific nodes e.g., deploy on ARM or x64 nodes but also be restricted to a subset of the resources on those nodes.

3.2 Scheduler & Orchestrator Architecture

Figure 2 shows the internal architecture and functions of the scheduler and physical & virtual resource orchestrator component integrated with the kubernetes control plane elements. The orchestrator receives a policy parameter as input at start-up time and spins up two concurrent processes namely the Policy function and the Resource Monitor function. The policy parameter determines the scheduling behavior and the placement of service VNFs on the worker nodes. Currently, two options are available for policy parameter targeting energy saving and load balancing. In the energy saving mode, the orchestrator targets running the least number of worker nodes in running state in order to conserve energy expended by the cluster. Practically, this implies a single worker node is kept running for as long as the deployed service VNFs are providing the required quality of service. A new worker node is boot up only when the running services are scaled beyond the resources of the running worker nodes or when new deployments are requested that either require more physical resources or are targeting a particular node/set of nodes in the cluster with specialized hardware. A service deployment may target a specific set of nodes for several reasons such as due to assigned quota, resource constraints or performance reasons. For example, a service might request a VNF to be deployment on a node that has SSD storage or a GPU for graphic processing. The master node keeps a record of the hardware attributes of the worker nodes in the cluster. The orchestrator can map these attributes to the requirements of a service deployment. The other option for policy parameter is load balancing which keeps all the worker nodes in running state and priorities minimal delay in service deployment and scaling latency. This is achieved by compromising on the overall energy consumption of the cluster. The whole service deployment and scheduling process is executed as follows:

- Service deployment is requested by providing a service template file to the Kubernetes API Server component. To do this, a service provider needs to authenticate its access with the API Server. The service template is provided as Yaml [3] file describing the service at a high abstraction level such as the required VNFs (Docker containers to run), hardware resources required, service and target node attributes.

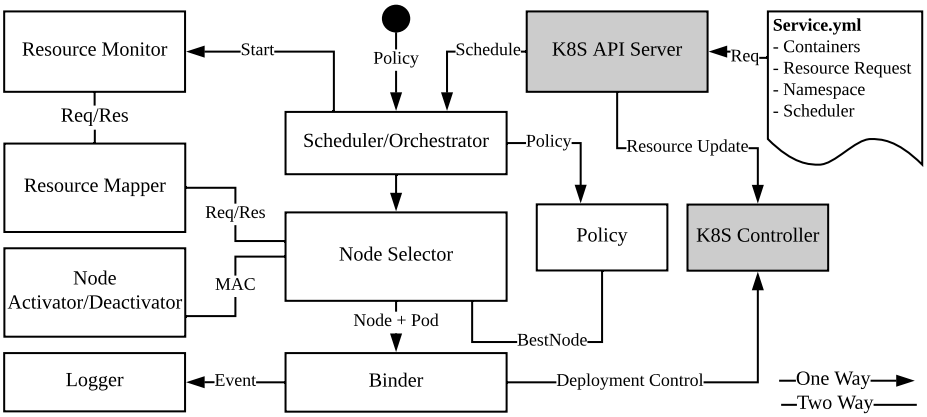

Fig. 2. Software architecture and functional diagram

- The API server receives the deployment requests and pushes a scheduling request trigger to the orchestrator component. At this point, the scheduling and orchestration component has to determine the criteria of VNF placement resource scaling. These include (a) find the worker nodes in the cluster that have the required resources available to host the VNFs requested by the deployment (b) determine, for each VNF, a single node out of the possibly multiple qualified nodes as host (c) determine the state of the node which can be either ready or switched off (d) bind the VNF to the selected node if the node is in ready state (in load balancing policy) or first boot the node (in energy saving policy) and then bind the VNF to the selected node once it is in ready state.
- For task (a), a Resource Mapper function takes the requested resources of a given service deployment request and compares them to the active resource map maintained by the Resource Monitor process instantiated by the scheduler. It should be noted that the Resource Monitor keeps a real-time load of running nodes only and an indication of the physical capacity of the nodes that are offline but part of the cluster. This way, the resource mapping process considers all nodes in the cluster instead of the online ones if the scheduling policy is load balancing or the VNFs image is for a particular hardware architecture. The resource mapper will return the list of all qualified nodes that can host the service VNFs.
- For task (b), the policy function is responsible for reducing the possible list of several qualified worker nodes to a single node based on the policy used by the scheduler. It is worth pointing out here that this process is sequential i.e., a deployment may contain several VNFs (containers) and the deployment of those are carried out sequentially in order to fulfill the load balancing or energy saving policy requirements.

– For task (c), the node activator module takes care of booting up a node based on the node ID in case it is offline. It uses the Wake On Lan feature of NICs on the worker nodes.
– Finally for task (d), the binder function does the actual scheduling of the VNF on the selected node. For this, the binder interacts with the kubernetes control plane elements to download the VNF container image (if not present locally), and then boot it up with the required resources provisioned. Lastly the Binder function logs the success and failure events of the binding process.

4 Evaluation

We have carried out preliminary evaluation of the resource consumption (Storage, Memory, CPU) of the control elements in our prototype edge cloud environment and its performance in terms of service deployment and deletion latencies. Figure 3 shows the image sizes of Kubernetes core control elements and our add-on purpose-built scheduler and orchestrator (labeled as Sch+Orch). The add-on component has been implemented in Golang and compiled into a static binary. As evident from figure 3, the container image sizes of the core Kubernetes elements are small enough to be deployed on tiny storage resource constrained devices. Second, it is very easy to create add-on elements for specific functionality such as Sch+Orch that has our statically linked binary embedded inside a tiny sized Docker image (5.3MB only).

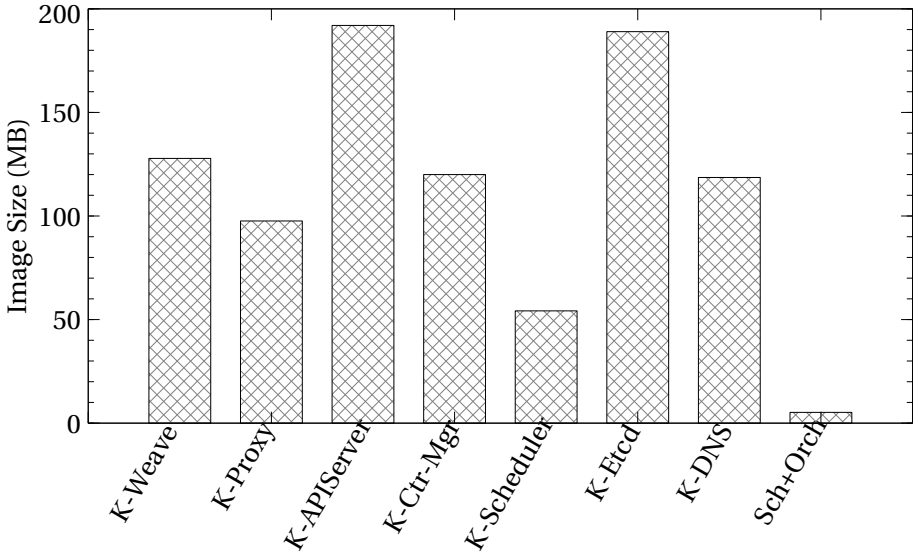

Fig. 3. Container image sizes for core control elements

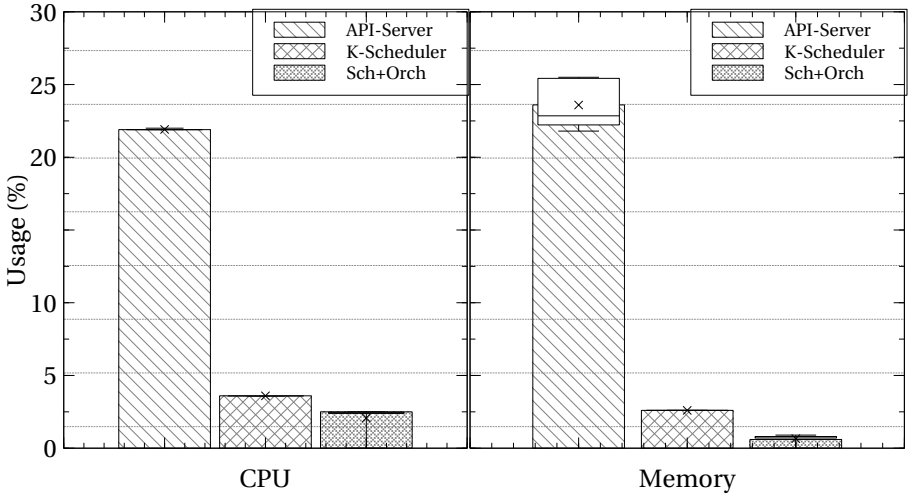

Fig. 4. CPU and Memory utilization of core control elements

Figure 4 shows the CPU and Memory utilization on the Master node (Raspberry Pi 3) of the three main control elements that are used during deployment of a service containing two VNFs. In general, all elements have a very small resource utilization footprint during an actual service deployment. However, since our custom Sch+Orch element takes away the scheduling and orchestration duties from the Kubernetes default scheduler, there is no observable variance in its CPU and Memory consumption during the deployment. In fact, the only significant variance is in the memory footprint of the API server and the CPU utilization of our scheduler VNF. This is expected since a service deployment is an active interaction process between the Kubernetes API Server and the scheduler VNF. To evaluate the scheduling and service deployment capabilities of our platform and to analyze the effects of scheduling policies on the deployment latencies, we deployed a couple of prototype web and video streaming edge services in our evaluation platform. Figure 5 shows the two prototype services and their VNF composition. On the left side, a service that combines web services and access services is depicted presented. This service targets the ARM nodes in the cluster as the Docker images for Nginx VNF and WiFi access point VNFs have been developed for that platform. The association between VNF architecture and the nodes having that physical architecture is done by label matching between the service template (yaml file) and the labeled nodes. On the right side in figure 5, a mixed architecture streaming service is depicted. This service template exemplifies a video streaming scenario where users of different privileges are provisioned video streaming service of different quality. The VNFs in this service includes an access VNF (WiFi access) and two streaming VNFs that stream video files in different quality. The video files are provided to both streaming services by a back-end video store VNF. To eliminate some of the latency fac-

tors of service deployment, all the Docker images for required service VNFs were downloaded prior to the actual deployment. Figure 6 shows the service deploy-

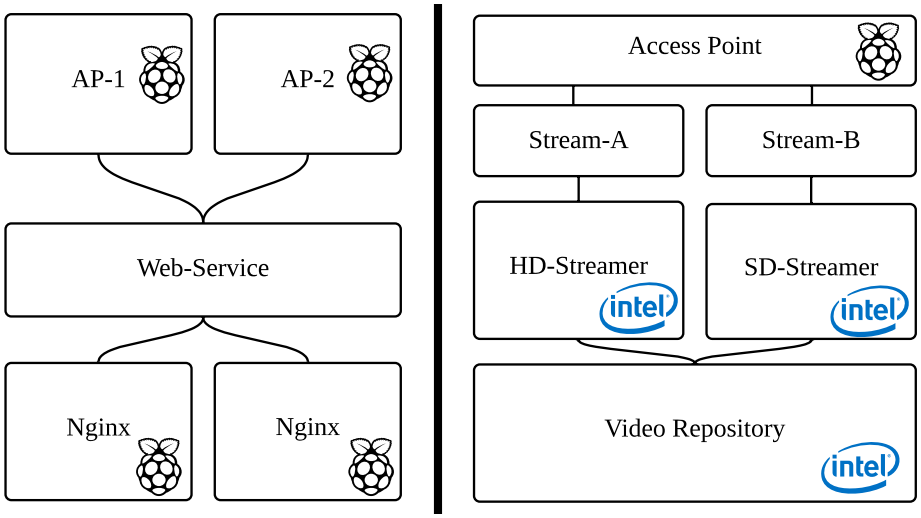

Fig. 5. Prototype edge services for web hosting and video streaming

ment and deletion time for the two services per the load-balancing and energy saving scheduling policies. The web-service is entirely hosted on ARM nodes and uses the load-balancing scheduling policy. As evident, the service takes approximately 26 seconds to be active in the edge cloud including providing the WiFi Access and the Nginx web-server. The video streaming service is deployed on different architecture nodes where most of the streaming VNFs are deployed on x64 machines while the WiFi access VNF is hosted on ARM node. This service uses the energy efficient scheduling policy where the x64 machine is first booted up from an OFF state before the streaming VNFs can be deployed. The x64 node takes around 50 seconds to boot up and associate with the Master node. The load balancing and energy saving policies can be mixed depending on the service requirements. Figure 6 also shows the respective deletion times for the two example edge services.

5 Conclusion

In this paper, we characterized the core attributes of network edge and its implications for brining cloud abstraction and SDN/NFV benefits to the end-users. We also presented a candidate architecture for heterogeneous edge clouds supporting multi-tenant edge services. Moreover, a prototype VNF scheduler and joint physical virtual resource orchestrator was presented and evaluated for prototype edge service deployment. In future we intend to extend the orchestration

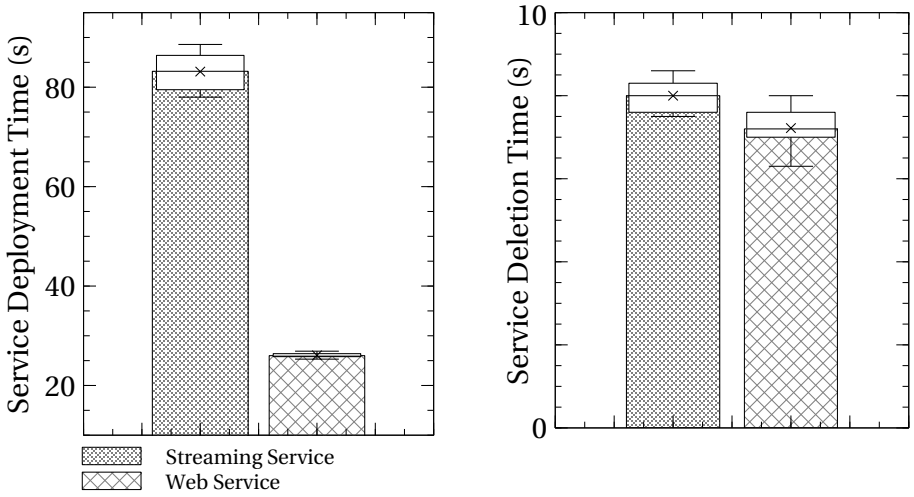

Fig. 6. Service deployment and deletion latencies

umbrella across multiple distributed edge-clouds by tying together a federated cloud environment. We also intend to investigate the interplay between self-organizing edge services and resource orchestration in host infrastructure using Artificial Intelligence algorithms.

References

1. CISCO: Cisco Visual Networking Index: Forecast and Methodology, 2016-2021 [Online] https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html
2. Kubernetes: Production-Grade Container Orchestration System, 2018 [Online] https://www.kubernetes.io
3. YAML: Yaml ain't markup language. [Online] http://yaml.org/
4. ETSI: Network Functions Virtualisation (NFV) Management and Orchestration, Technical Report. 2014 [Online] http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf
5. OSM: OpenSource Management & Network Orchestration 2018 [Online] https://osm.etsi.org/
6. OpenStack: Open source software for creating private and public clouds 2018 [Online] https://www.openstack.org/
7. CORD: Central Office Re-architected as a Datacenter 2018 [Online] https://opencord.org/
8. Blenk, A., Basta, A., Reisslein, M., Kellerer, W.: Survey on Network Virtualization Hypervisors for Software Defined Networking. IEEE Communications Surveys & Tutorials, vol. 18, no. 1, 655-685, (2016)