Automation of Virtualized 5G Infrastructure Using Mosaic 5G Operator over Kubernetes Supporting Network Slicing

1st Fandi A. Wiranata
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
fandi.z.w@gmail.com

2nd Wervyan Shalannanda
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
wervyan@stei.itb.ac.id

3rd Rahmat Mulyawan
PUI-PT Mikroelektronika
Bandung Institute of Technology
Bandung, Indonesia
rahmat@std.stei.itb.ac.id

4th Trio Adiono
University Center of Excellence on Microelectronics
Bandung Institute of Technology
Bandung, Indonesia
tadiono@stei.itb.ac.id

***Abstract*—The rising of diverse requirements and the traffic explosion lead to many challenges for traditional mobile network architecture on flexibility, scalability, and deployability. Service-based architecture is introduced into mobile networks to meet new requirements in the 5G era. The monolithic network elements are split into smaller network functions to provide customized services. However, the management and deployment of network function in service-based 5G core networks are still big challenges. In this paper, we propose the network automation using Kubernetes for automating application deployment, while using Openshift Operator as a tool to manage 5G services. This operator lets us deploy all Mosaic 5G inside pods. In this Mosaic 5G Operator, especially in custom resource, we define the composition of core network, Radio Access Network (RAN), FlexRAN, ElasticSearch, and Kibana (as data visualization). For this purpose, we use the containerized OpenAirInterface (OAI) to deploy and demonstrate the automatability with extensive slicing radio support.**

***Index Terms*—Mosaic 5G, Kubernetes, OAI, Virtualization, 5G**

I. Introduction

In recent years, both in types of communication and traffic are experiencing unprecedented changes in the mobile network. Video traffic grows fast which caused increasing of data traffic that is generated in the mobile network. Cisco [1] notes that 47 percent of mobile data traffic will increase at compound annual growth rate (CAGR) from 2016 to 2021, reaching 49 exabytes per month by 2021. Meanwhile, 780 million (2016) to 3.3 billion (2021) of M2M (machine to machine)

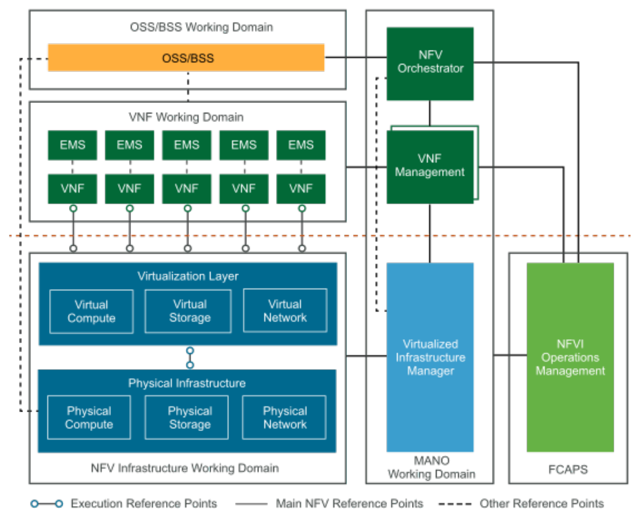


Fig. 1: NFV reference architecture framework [3].

connections will grow fast and have three scenarios in the 5G era include, i) Enhanced Mobile Broadband (eMBB) focuses on services characterized by high data rates, i.e. high definition (HD) videos, virtual reality (VR), augmented reality (AR), etc. ii) Ultra-Reliable and Low Latency Communications (URLLC) provide latency-sensitive services, i.e. self-driving. iii) Massive Internet of Things (mIoT) consists of devices in a huge connection density, i.e. smart city, smart agriculture, etc. [2]. Because of that, the requirements and characteristics of traffic in mobile networks are evolving to more complex.

978-1-7281-7598-0/20/$31.00 ©2020 IEEE

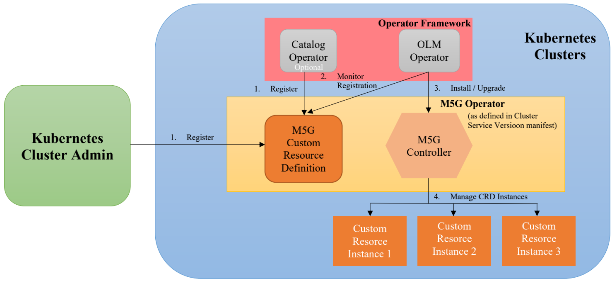


Fig. 2: Basic Mosaic 5G Operator on Kubernetes.

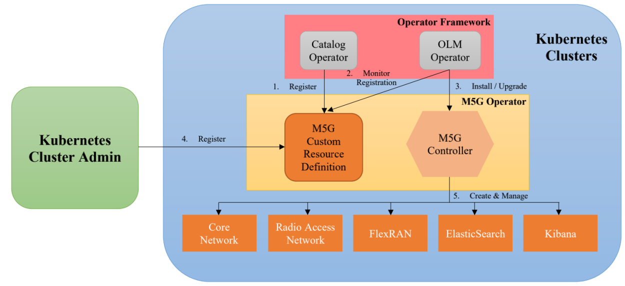


Fig. 3: Supporting platforms on Mosaic 5G Operator.

To realize this diversity or many use cases, we must implement a form of 5G architecture called network slicing. In network slicing, Software Defined Networking (SDN) and Network Functions Virtualization (NFV) are fundamentals. Network slicing is a form of virtual network architecture that allows multiple virtual networks to be created on top of a common shared physical infrastructure using the same principles behind SDN and NFV in fixed networks. SDN and NFV are deployed to deliver the greater network flexibility by allowing traditional network architectures to be partitioned into virtual elements that can be linked.

ETSI defines an architecture framework of NFV that shown in Fig. 1. The figure consists of several functional blocks such as NFV Infrastructure (NFVI), Element Management System (EMS), Virtualized Network Function (VNF), Operations and Business Support Systems (OSS/BSS), and NFV management and orchestration (MANO). VNF can virtualize network function, e.g. RAN and Evolved Packet Core (EPC). One or several VNFs are managed by EMS. NFV MANO can be applied to deploy the virtualized 5G infrastructure. Based on Fig. 1, there are three functional blocks of NFV MANO that are considered as : i) NFV Orchestrator (NFVO) is responsible of network slice lifecycle management with the VNF lifecycle and the NFVI resources (JOX [6], Open Source MANO (OSM) [7], and Open Baton [8]). ii) Virtual Network and Function Manager (VNFM) take care of deploying, scaling, monitoring, and removing VNFs on a VIM (e.g. Juju). iii) Virtualized Infrastructure Manager (VIM) is in charge of managing and controlling the NFVI and abstract them into the physical hardware resources of computing, network, and storage (e.g. OpenStack [4], CloudStack, Google Kubernetes VIM, etc).

Virtualized 5G infrastructure [11] can be realized with one of the approaches, which is SliceNet[10]. Two egalitarian parts of network slicing are a slice of the core network (CN) and a slice of the Radio Access Network (RAN). Furthermore, evolved packet core (EPC) bundle together of virtual resources and physical. The different levels of isolation and sharing are provided by RAN slicing according to the slice requirement, which can be more satisfied with different service level agreements in 5G [9]. In [11], the authors leveraging OpenAirInterface (OAI) [12] and Mosaic-5G [13– 15] to realize the concept of network slicing. Softwarizing mobile network functions from the access network to the EPC of the mobile network can be developed using OAI which is an open-source project. The EPC software and the access network software are two parts of OAI as known as OAI-CN and OAI-RAN, respectively. The OAI-CN consists of the Mobility Management Entity (MME), Home Subscriber Server (HSS), the Serving and the Packet Data Network Gateway (S-GW and P-GW). The OAI-RAN is a form of Base Station with a Remote Radio Unit (RRU) and Baseband Unit (BBU). Standard Linux-based computing equipment can be used to deploy OAI. Mosaic-5G is also open-source project based that has three main platforms: i) FlexRAN enabling monitoring, control, programmability in the RAN domain. ii) LL-MEC acts as a controller for edge and core domains providing a subset of features as specified by ETSI MEC. iii) JOX is an event-driven juju-based service orchestrator core to interact with different network domains.

In this paper, we explain the detail of orchestration part and we propose the deployment of virtualized 5G infrastructure on container orchestrated by Kubernetes and Mosaic 5G Operator with extensive slicing radio support. Specifically, for automating application deployment, we use Kubernetes and Openshift Operator as a tool to manage 5G services and our implementation of the FlexRAN use case that supports slicing.

This paper consists of several sections. In section II we show the Mosaic 5G Operator architecture in detail. In section III we explain our contributions in detail where we propose an architecture and orchestration flow achieved in 5G dynamic network automation and management by Kubernetes and Mosaic 5G Operator. In section IV we describe the results of implementing the 5G Virtualized Infrastructure on Kubernetes. In section V we conclude our work and highlight some important parts.

II. Mosaic 5G Operator Architecture

We use Operator to manage the lifecycle of applications and fully automate the deployment under Kubernetes. It's more portable and has better integration since Kubernetes as a service can run the Operator. The operator also able

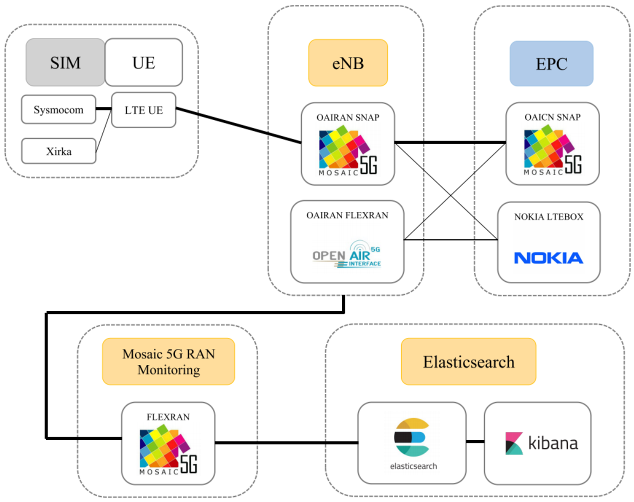


Fig. 4: The functional blocks of Mosaic 5G Operator.

to manage complex services, which is by administrators and their operation scripts, e.g. ansible.

Fig. 2 shows functional blocks of Mosaic 5G Operator. The functional blocks include : i) Catalog Operator, it is responsible for resolving and installing Cluster Service Version (CSVs). A CSV is composed of Custom Resource Definitions (CRD). ii) Operator Lifecycle Manager (OLM) helps users install, update, and manage the lifecycle of all Operators and their associated services running across their clusters.

The controller, which is the core concept of Kubernetes, is utilized by Operator to achieve its ability,. On the master node, the controller continuously loops and listens for any changes to its pods. If the controller detects these changes, then the controller will perform the corresponding actions. An Openshift Operator SDK [17], dubbed as Mosaic 5G Operator is implemented using Golang, where supports service deployment, reconfiguration, and upgrading/downgrading of service's version. More details, We implement Mosaic 5G platforms (i.e. FlexRAN and OAI), ElasticSearch, and Kibana in Custom Resource shown in Fig. 3.

III. Mosaic 5G on Kubernetes

OAI is utilized by Mosaic 5G to virtualize and implement 5G network functions, which provide more flexible service in different applications. Fig. 4 illustrates the overall architecture that we already developed, both EPC and eNB supports the functions of CN and RAN, respectively. The OAI-RAN Snap and OAIRAN FlexRAN are integrated into eNB while OAI-RAN Snap and Nokia LTEBox are integrated into EPC for the LTE network. We could choose which eNB, EPC, and other supporting components only one line. On the other hand, before the mobile network runs on top of commodity hardware, the functionalities are disaggregated via virtualization. Therefore, the functions of CN and RAN can be disaggregated.

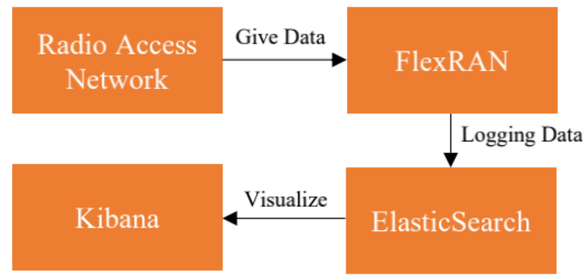


Fig. 5: Data visualization schemes.

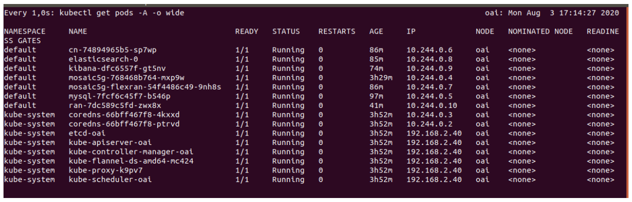
Under this condition, virtualized CN can cooperate with virtualized RAN to support E2E network slicing aiming to adapt network management resources and provide more efficiency 5G services.

Based on Fig. 4, we first install Kubernetes environtment and then integrate the eNB and EPC with Mosaic 5G Operator to realize the VRAN-CN infrastructure. We create a script to support slicing between OAI-RAN Snap & OAI-CN Snap, OAIRAN FlexRAN & OAI-CN Snap, and OAI-RAN Snap & Nokia LTEBox to implement the procedure, description in Table 1.

TABLE I: Mosaic 5G (M5G) on Kubernetes Implementation Procedure.

	Procedure	Command
1.	Build Kubernetes environtment	$ ./createk8s.sh start flannel (start kubernetes flannel)
2.	Apply M5G Operator CRD to Kubernetes	$ ./m5goperator.sh init
3.	Start M5G as a container/pod in Kubernetes	$ ./m5goperator.sh container start
4.	Deploy cluster role for api access	$ ./api.sh init
5.	Deploy OAI Container/Pod	#choose what you want to deploy
		$ ./api.sh apply_cr #to use OAI-RAN Snap & OAI-CN Snap $ ./api.sh apply_cr_slicing #to use OAIRAN FlexRAN & OAI-CN Snap $ ./api.sh apply_cr_ltebox #to use OAI-RAN Snap & Nokia LTEBox
6.	Check all running pods and its IP address	$ ./m5goperator.sh watch_pods

Fig. 4 also shows other functional blocks of Mosaic 5G Operator, which is Mosaic 5G RAN monitoring and ElasticSearch. Furthermore, we can monitor the network and visualize the data, as depicted in Fig. 5. This process can be observed in Fig. 5, ElasticSearch will get log file from FlexRAN and save it, and we can use Kibana to visualize it.



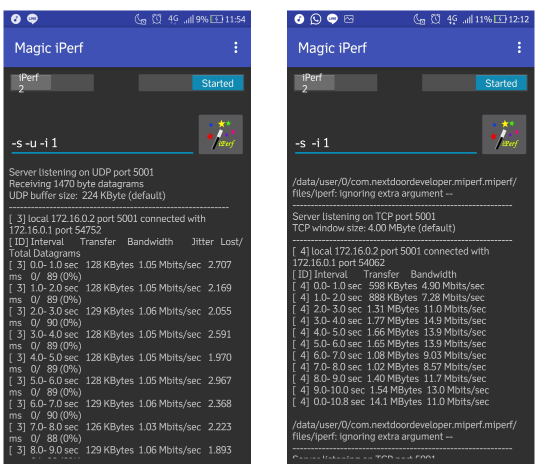
```
Every 1,0s: kubectl get pods -A -o wide                                                      oai: Mon Aug  3 17:14:27 2020

NAMESPACE     NAME                                 READY   STATUS    RESTARTS   AGE     IP             NODE   NOMINATED NODE   READINE
SS GATES
default       cn-74894965b5-sp7wp                  1/1     Running   0          86m     10.244.0.6     oai    <none>           <none>
default       elasticsearch-0                      1/1     Running   0          85m     10.244.0.8     oai    <none>           <none>
default       kibana-dfc6557f-gt5nv                1/1     Running   0          74m     10.244.0.9     oai    <none>           <none>
default       mosaic5g-768468b764-mxp9w            1/1     Running   0          3h29m   10.244.0.4     oai    <none>           <none>
default       mosaic5g-flexran-54f4486c49-9nh8s    1/1     Running   0          86m     10.244.0.7     oai    <none>           <none>
default       mysql-7fcf6c45f7-b546p               1/1     Running   0          97m     10.244.0.5     oai    <none>           <none>
default       ran-7dc589c5fd-zwx8x                 1/1     Running   0          41m     10.244.0.10    oai    <none>           <none>
kube-system   coredns-66bff467f8-4kxxd             1/1     Running   0          3h52m   10.244.0.3     oai    <none>           <none>
kube-system   coredns-66bff467f8-ptrvd             1/1     Running   0          3h52m   10.244.0.2     oai    <none>           <none>
kube-system   etcd-oai                             1/1     Running   0          3h52m   192.168.2.40   oai    <none>           <none>
kube-system   kube-apiserver-oai                   1/1     Running   0          3h52m   192.168.2.40   oai    <none>           <none>
kube-system   kube-controller-manager-oai          1/1     Running   0          3h52m   192.168.2.40   oai    <none>           <none>
kube-system   kube-flannel-ds-amd64-mc424          1/1     Running   0          3h52m   192.168.2.40   oai    <none>           <none>
kube-system   kube-proxy-k9pv7                     1/1     Running   0          3h52m   192.168.2.40   oai    <none>           <none>
kube-system   kube-scheduler-oai                   1/1     Running   0          3h52m   192.168.2.40   oai    <none>           <none>
```

Fig. 6: Illustration of Mosaic 5G pods that successfully deployed on Kubernetes.

IV. Deployment of Virtualized 5G Infrastructure on Kubernetes with Mosaic 5G Operator

This section describes the results of implementing the 5G virtualized infrastructures on Kubernetes. In our implementation, we create a script to assist the configuration on Kubernetes which can support the automation deployment of Virtualized 5G Infrastructure. Fig. 6 shows all running pods and its IP address that we have already developed. Specifically, the RAN and CN are implemented by OAI (infrastructure) and Mosaic 5G Operator integrates the virtualized functions. We use Sysmocom SIM card programmable (sysmoUSIM-SJS1) to connect UE to OAI Network. OAI by default saves user parameters in user database that is located in HSS. SIM card is used to authenticate UE that wants to connect to network. If authentication process is successful, UE can attach to network. Authentication successful if parameter in SIM card is match with parameter in HSS. Because sysmoUSIM-SJS1 card is programmable, we can change parameters inside it.

A. UDP and TCP Testing



(a) UDP. (b) TCP.

Fig. 7: The phone (server) output.

We create iperf server for UDP and TCP testing and send report with 1 second interval shown in Fig. 7. The iperf as a client and send packets to the designated IP in UDP and TCP mode, respectively. Fig. 8 (a) shows in 10 seconds packets can be sent using 1 Mbps bandwidth and succesfully send all 893 packets. The jitter that we have in this connection from the phone is 2.139 ms. We can observe in UDP testing the bandwidth that is used in default is 1 Mbps, while in Fig. 8 (b) the maximum bandwidth avalaible is 15 Mbps. The reason is that in TCP testing is always try to use the maximum bandwidth and 15 Mbps is the highest, it could get in bandwidth.

```
oai@oai:~/mosaic5g$ iperf -c 172.16.0.2 -u
------------------------------------------------------------
Client connecting to 172.16.0.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 172.16.0.1 port 54752 connected with 172.16.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   2.139 ms    0/  893 (0%)
```

(a) UDP.

```
oai@oai:~/mosaic5g$ iperf -c 172.16.0.2
------------------------------------------------------------
Client connecting to 172.16.0.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 172.16.0.1 port 54062 connected with 172.16.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  14.1 MBytes  11.6 Mbits/sec
```

(b) TCP.

Fig. 8: The computer (client) output.

B. Slicing in RAN Testing

After connecting the UE, run the lifecycle bash and follow the output of the bash. The script will first decrease the first slice from 100 % to 20%. Fig. 9 shows that the percentage of the slice is decreasing. The output of FlexRAN will change from Fig. 9 (a) to Fig. 9 (b). Next the script create a slice with 50% percentage shown in Fig. 10 (a) and change DL slice shown in Fig. 10 (b).

```
...
"sliceConfig": {
        "dl": [
          {
            "id": 0,
            "label": "xMBB",
            "percentage": 100,
            "isolation": false,
            "priority": 10,
            "positionLow": 0,
            "positionHigh": 25,
            "maxmcs": 28,
            "sorting": [
              "CR_ROUND",
              "CR_SRB12",
              "CR_HOL",
              "CR_LC",
              "CR_CQI",
              "CR_LCP"
            ],
            "accounting": "POL_FAIR",
            "schedulerName": "schedule_ue_spec"
          }
...
```

```
...
"sliceConfig": {
        "dl": [
          {
            "id": 0,
            "label": "xMBB",
            "percentage": 20,
            "isolation": false,
            "priority": 10,
            "positionLow": 0,
            "positionHigh": 25,
            "maxmcs": 28,
            "sorting": [
              "CR_ROUND",
              "CR_SRB12",
              "CR_HOL",
              "CR_LC",
              "CR_CQI",
              "CR_LCP"
            ],
            "accounting": "POL_FAIR",
            "schedulerName": "schedule_ue_spec"
          }
...
```

(a) Percentage: 100%. (b) Percentage: 20%.

Fig. 9: The number of resource blocks that this slice is allowed to use from FlexRAN output.

Fig. 10 shows the two slices with id number 0 and 3, and the phone is connected to slice number 3 for the Downlink transmission. Disconnecting the phone will

```
...
 "sliceConfig": {
             "dl": [
               {
                 "id": 0,
                 "label": "xMBB",
                 "percentage": 20,
                 "isolation": false,
                 "priority": 10,
                 "positionLow": 0,
                 "positionHigh": 25,
                 "maxmcs": 28,
                 "sorting": [
                   "CR_ROUND",
                   "CR_SRB12",
                   "CR_HOL",
                   "CR_LC",
                   "CR_CQI",
                   "CR_LCP"
                 ],
                 "accounting": "POL_FAIR",
                 "schedulerName": "schedule_ue_spec"
               }
               {
                 "id": 3,
                 "label": "xMBB",
                 "percentage": 50,
                 "isolation": false,
                 "priority": 10,
                 "positionLow": 0,
                 "positionHigh": 25,
                 "maxmcs": 28,
                 "sorting": [
                   "CR_ROUND",
                   "CR_SRB12",
                   "CR_HOL",
                   "CR_LC",
                   "CR_CQI",
                   "CR_LCP"
                 ],
                 "accounting": "POL_FAIR",
                 "schedulerName": "schedule_ue_spec"
               }
...
```


(a) Percentage: 50%


```
...

"UE": {
       "ueConfig": [
         {
           "rnti": 24385,
           "timeAlignmentTimer": 7,
           "measGapConfigPattern": 4294967295,
           "measGapConfigSfOffset": 4294967295,
           "transmissionMode": 0,
           "ueAggregatedMaxBitrateUL": "0",
           "ueAggregatedMaxBitrateDL": "0",
           "capabilities": {
             "halfDuplex": 0,
             "intraSFHopping": 0,
             "type2Sb1": 1,
             "ueCategory": 4,
             "resAllocType1": 1
           },
           "ueTransmissionAntenna": 2,
           "ttiBundling": 0,
           "maxHARQTx": 4,
           "betaOffsetACKIndex": 0,
           "betaOffsetRIIndex": 0,
           "betaOffsetCQIIndex": 8,
           "ackNackSimultaneousTrans": 0,
           "simultaneousAckNackCqi": 0,
           "aperiodicCqiRepMode": 3,
           "tddAckNackFeedback": 4294967295,
           "ackNackRepetitionFactor": 0,
           "extendedBsrSize": 4294967295,
           "imsi": "208930000000001",
           "dlSliceId": 3,
           "ulSliceId": 0,
...
```


(b) Slice number 3 for the Downlink transmission.

Fig. 10: The output from FlexRAN.

cause the script remove the second slice and restore the bandwidth percentage to the first one.

V. Conclusion

In this paper, we proposed the network automation, which integrates multiple components, i.e. Kubernetes, Mosaic 5G, OAI, Openshift Operator SDK, ElasticSearch, Kibana, and Docker, from the infrastructure until the orchestration level to build an E2E mobile network. Specifically, this paper addresses the concept of Mosaic 5G and demonstrates the integration of Mosaic 5G Operator over Kubernetes, which supports network slicing by setting the users into two created slices and performing radio resources allocation and validation between the two concurrent slices and inside the same slice. From this slice test we can then test to create multiple slices and configure the bandwidth for each slice.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021," Cisco White Paper, 2017.

[2] 3GPP, "5g: A tutorial overview of standards, trials, challenges, deployment and practice," Tech nical Report TR38.913, 2016.

[3] ETSI GS NFV 002 V1.1.1, "Network Functions Virtualisation (NFV); Architectural Framework ", Oct. 2013, [Online]. Available online at http://www.etsi.org/deliver/etsi gs/NFV/001 099/002/01.01.01 60/ gs NFV002v010101p.pdf

[4] OpenStack Foundation Report, "Accelerating NFV Delivery with OpenStack Global Telecoms Align Around Open-Source Networking Future", 2016, [Online]. Available: https://www.openstack.org/assets/telecomsand-nfv/OpenStack-Foundation-NFV-Report.pdf.

[5] Juju, Aug., 2020, [Online]. Available: https://jujucharms.com/

[6] K. Katsalis and others, "JOX: An Event-ariven Orchestrator for 5G Network Slicing", in Proc. IEEE/IFIP Network Operations and Management Symposium, pp. 1-9, August. 2020.

[7] "Open Source MANO", Aug., 2020, [Online]. Available: https://osm.etsi.org/

[8] "Open Baton", Aug., 2020, [Online]. Available: https://openbaton.github.io/

[9] C.-Y. Chang and N. Nikaein, "RAN Runtime Slicing System for Flexible and Dynamic Service Execution Environment", IEEE Access, vol. 6, pp. 34018-34042, August. 2020.

[10] "SliceNet, Deliverable 3.1 - Design and Prototyping of SliceNet Virtualized Mobile Edge Computing Infrastructure", August, 2020, [Online]. Available: https://slicenet.eu/deliverables/

[11] N. Nikaein and others, "SliceNet, Deliverable 3.1 - Design and Prototyping of SliceNet Virtualised 5G RAN-Core Infrastructure", August, 2020, [Online]. Available: https://slicenet.eu/deliverables

[12] N. Nikaein and others, "OpenAirInterface: A Flexible Platform for 5G Research", in ACM SIGCOMM Computer Communication Review, vol. 44, no. 5, pp. 33-38, Oct. 2014.

[13] "Mosaic5G.io", Aug., 2020, [Online]. Available: http://mosaic-5g.io/

[14] N. Navid, "Mosaic5G - Agile network service delivery platforms", in 4th OpenAirInterface Workshop, Nov. 2017, [On line]. Available: http://www.eurecom.fr/en/publication/5368/detail/mosaic5g-agilenetwork-service-delivery-platforms

[15] N. Navid et al, "Mosaic5G - Agile network service delivery platforms for 5G Research", in ACM SIGCOMM Computer Communication Review, vol. 48, pp. 1-6, Jul. 2020, [Online]. Ava ilable: http://www.eurecom.fr/en/publication/5609/download/comsyspubli-5609.pdf

[16] OpenAirInterface, August. 2020, [Online]. Available: http://www.openairinterface.org

[17] Operator SDK, July, 2020, [Online]. Available: https://github.com/operator-framework/operator-sdk

[18] Redhat, "Understanding Operators." [Online]. Available: https://docs.openshift.com/container-platform/4.2/operators/olmwhat-operators-are.html