Connecting Remote eNB with Containerized 5G C-RANs in OpenStack Cloud

Bruno Dzogovic
Oslo Metropolitan University
Oslo, Norway
bruno@oslomet.no

Bernardo Santos
Oslo Metropolitan University
Oslo, Norway
bersan@oslomet.no

Van Thuan Do
Wolffia AS
Oslo, Norway
vt.do@wolffia.net

Boning Feng
Oslo Metropolitan University
Oslo, Norway
boningf@oslomet.no

Niels Jacot
Wolffia AS
Helsinki, Finland
n.jacot@wolffia.net

Thanh van Do
Telenor & Oslo Metropolitan University
Fornebu, Norway
thanh-van.do@telenor.com

***Abstract* - Cloud-Radio Access Networks are one of the main enablers for driving the 5G technology. They allow creation and utilization of core network components in a precise flexible manner that implies of the possibility for resource redistribution across different geographical regions, with reduced operator costs. With virtualization and frontend RF functionality splitting, the key processing migrates to the cloud, which opens a wide palette of facets for determining manifold layers of security and operability for the overall 5G access. Due to the familiarity with the specific concerns of IoT devices' vulnerability and security apprehensions, we provide an initial testbed for mitigating the lower-layer problems, which entails establishment of a specific network function that communicates with the OpenStack Neutron and integrates with the Keystone service. Furthermore, we introduce an approach for federating the Keystone service with OpenID Connect, enabling access to different network slices at the backhaul 5G network.**

Keywords—5G, C-RAN, Kuryr, Docker, SR-IOV, IdP, IoT, OpenStack

I. Introduction

One of the main objectives of the 5G mobile network is to support efficiently diverse Internet of Things (IoT) applications also called IoT verticals and the focus has been to fulfil a wide range of quality of service requirements. However, the current most popular wireless technologies for IoT are Wireless LAN (WLAN) [1], ZigBee [2] or Bluetooth Low Energy (BLE) [3] and to become the wireless technology of choice for IoT, 5G will need to be capable to provide higher level of security. In a virtualized and cloudified 5G network, one of the weakness point is the connection between the remote radio unit (RRU) and the edge cloud. To address this critical security issues, it is essential to strengthen the communication link between the remote radio frontend and the edge/cloud location. For that purpose, it is important to focus on the underlying network setup and address issues related to the networking between containers deployed in the cloud and bare-metal servers. One popular technique that utilizes traffic encryption is tunneling, which can be useful for avoidance of complex network traversals such as through MPLS (Multi-Protocol Label Switching) that carries multiplexed variety of traffic and is exceptionally convoluted to administer. Tunneling can be expedient in cases of establishing end-to-end communication via encrypted tunnels, as in the case of VPN where the two sites need to explicitly share private connection. This procedure can be exorbitant to instigate and unnecessary in many unpretentious scenarios, especially in bandwidth-restricted networking scenarios as well as low-latency bound networks.

To provide alternative manner of connection to the tunneling method, while preserving flatter network topology, the OpenStack cloud can be complemented with specific networking plugins that can allow containerized applications to connect via remote regions as part of the underlying physical network fabrics. Therefore, in this paper we exemplify the communication to the virtualized C-RAN network, namely the remote containerized cloud EPC (Evolved Packet Core), BBU (Baseband Unit) and a RRU (Remote Radio Unit) that are split functional parts of the eNB (evolved Node-B), communicating within the physical network underlay. A conclusion is thus provided, with the emphasis of the requirements for establishing an identity federation to enable integration of variety of IoT devices in such setup.

II. Related Work

In order to adopt a practicable approach for realizing the secure platform, the open-source paradigm is applied. The wide range of open-source assemblies allow establishment of a simplified solution that can be a further subject of automation and automated deployment, as well as to cloud-

 DOI: http://dx.doi.org/10.1109/CSCloud/EdgeCom.2019.00013

oriented services while retaining the best DevOps practices. This not only allows attainable fundamental configuration, but also paves the way for introducing 5G deployments in public and shared clouds.

A. *OpenAirInterface5G*

The primary mobile networking platform used in the experiment and has a wide range of possible establishments, is the OpenAirInterface (OAI) by Eurecom [1], an open-source virtual mobile networking system based on 4G EPC and eNB base-station function as a radio frontend (RF).

B. *OpenStack*

Providing a cloud environment for the deployed core network is the essential part of establishing a basis for 5G network initialization. As an open-source solution, we use the OpenStack cloud environment [5] as a public cloud at the Oslo Metropolitan University (see Figure 2). The operating system on which the services run is Ubuntu 18.04 with low-latency kernel.

C. *Docker container virtualization*

Another principal tool used for container virtualization of the OAI in OpenStack is Docker [6], a popular DevOps mechanism for packing applications into containers and running on a single host, whether it is a physical or a virtual machine on a bare metal server or cloud environment. One of the shortcomings of the Docker networking structure is the utilization of VXLAN, layer-2 virtual bridges and other network overlays. While the benefit of such methodology is simplicity and ease of deployment, the exclusive disadvantages are the cumulative network bottlenecking, overhead and latency that are undesirable for URLLC slice (Ultra Reliable Low Latency) deployments within the 5G networks.

Consequently, Docker establishes an operational image of the CN, deployed in a single-frame container or a container cluster with an orchestrator such as Kubernetes [7]. As represented in Figure 1, The single CN container implements the complete EPC, with the MME, S/PGW and HSS database together linked with multiple virtual interfaces and bridges (depending on the network driver used by Docker).

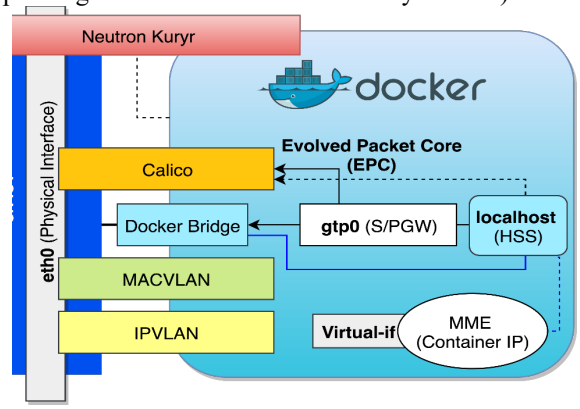

Figure 1. Docker Networking Drivers

In a clustered deployment, each of the EPC constituents can be commenced as separate containers in a uniform namespace, scaled accordingly to the load requirements or dissected to provide isolation if such necessities transpire. For the sake of simplicity, we use the single-container EPC and with different Docker networking drivers, such as Calico or MACVLAN. The disadvantages of this approach are the image size, which can reach ~10GB and the inability to granularly manipulate with the HSS database locally. For clustering the database, one needs to commission an external container service cluster for that purpose.

D. *Docker Open vSwitch Integration*

For the evasion of tunneling, the Docker daemon needs to link directly with the underlying cloud networking service, which is Neutron in OpenStack [8]. As Neutron is a very complex system, reduction of the complexity in containerized environments can yield positive outcomes. One plugin that offers solution for that scenario, developed by the OpenStack community, is the Kuryr network plugin for Neutron [9] that merges the functionality of the Docker daemon with the Neutron. This way, a Docker container can become part of a self-initialized Kuryr network that runs under the Docker instance and under the virtual machine on which it performs, close to the Neutron service. By applying uWSGI [10] python libraries, Kuryr authenticates its service with the Neutron via the OpenStack Keystone, after which the container can perform networking administrative tasks, as much as the specific user is allowed to.

The strongest characteristic of Kuryr is the possibility to configure the system to use SR-IOV (Single Root – Input/Output Virtualization) interfaces. The SR-IOV specification defines a standardized mechanism to virtualize PCIe devices. This mechanism can virtualize a single PCIe Ethernet controller to appear as multiple PCIe devices. Each device can be directly assigned to an instance, bypassing the virtual switch layer. As a result, users are able to achieve low latency and near-line wire speed [11].

E. *Identity Provision and Management*

As it is foreseen that the inclusion of **IoT** devices will achieve substantial magnitudes once the deployments of 5G networks are concluded, there are concerns regarding the lack of proper security measures for both the connection and the usage of the network. It is crucial to establish a security mechanism that uses the versatility and flexibility of Identity and Access Management (**IAM**) systems, which can be used in the upcoming fifth generation of the cellular network, thus an identity federation with provision and management resources is a solution that can facilitate all security operations needed for all devices [12]. This federation aims to achieve a common ground on how to identify connecting devices that are attempting to access a specific network slice, whether with the usage of SIM card and its attributes or with the device's attributes when no SIM is available. With that, it is possible to provide an identity to each device so that it can be authenticated into the network and use its resources.

In order to enable such authentication feature, the device should register in the network through its SIM card or by "piggybacking" through a gateway. This inclines on the necessity to reach the network's HSS database to check whether a matching record with the one that is trying to authenticate exists. Nevertheless, this access to the component, or rather the access to the relevant data must be assembled in a way that no compromise nor data exposure can happen in any of the layers (Network and Application).

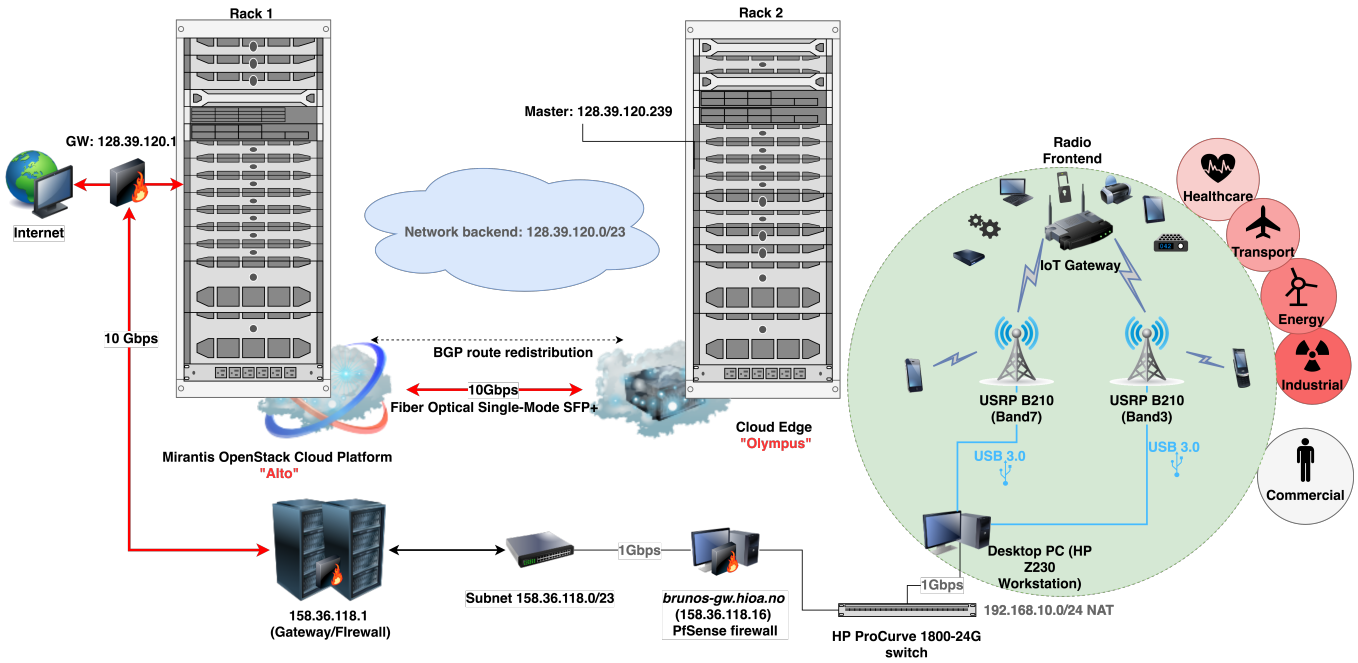

Figure 2. 5G4IoT Lab Infrastructure at the Oslo Metropolitan University

III. INFRASTRUCTURE DESIGN

With reference to the infrastructure depicted in Figure 2, several modes for tunnel networking and arrangement of the underlying network fabrics can be hitherto utilized: GRE encapsulation [13], VPN tunneling [14], L2TP [15], IPsec [16], PPTP [17], or GRE with minimal encapsulation for mobile networks within IP (defined in the RFC2004 standard) [18]. However, all of these approaches perform in network overlay mode, introducing overhead either from Ethernet broadcast frames and/or the tunneling encapsulation itself. The tunneling method is required for the reason that the underlying networking segment in the OpenStack Mitaka [19] cloud version is Open vSwitch-based Neutron and the layer-3 routing methods can be challenging to achieve without accumulation of surplus overhead and perplexing the virtual network segment. To circumvent the stacking of layer-2 network entities such as bridges, the deployed Virtual Machines (VMs) running the Evolved Packet Core (EPC) needs to be closely integrated with the Docker container environment in which the EPC is executed. Therein the Kuryr plugin for Docker integration with Open vSwitch [9]. Another reason why the VMs running in OpenStack require close integration with OvS is that there is no existent implementation for viable SCTP protocol communication in OpenStack Mitaka, which is essential for the communication between the EPC and the end User Equipment (UE), i.e. a smartphone. This viability exists after the introduction of the OpenStack "Pike" version [20]. Additionally, in the newer OpenStack versions, a different approach can be employed where a Calico Layer-3 BGP networking is supported and that can allow the Docker containers to become part of the underlying network fabrics, redistributing BGP routes between various Edges or POPs (Points of Presence) and MPLS networks. Accordingly, this enables interworking with third party SDN controllers such as OpenDaylight [21] [22].

With that in mind, we introduce alternatives and interworking for different platform versions while enforcing SLAs (Service Level Agreements) on different layers, priming the way for network slicing and providing Quality of Service with abstracted security. The necessity for interworking of the virtualized environment with the underlying network structures, prunes further towards the requirements for setup splits according to the 5G specifications. Certainly, the deployment of the overall 5G network is performed according to the 5GPP specification for the 5G architecture, using the OpenAirInterface (OAI) software [4] [25].

As previously stated, the OAI has the ability to establish a full 4G-based EPC, with virtualizing all the necessary objects for networking (MME, S/PGW, HSS) [26]. In addition to that, the functionality of the eNB (evolved Node-B) can be split according to the diverse modes of operation, explicated within the 3GPP specifications and in dependence on the requirements and expected goals. For that purpose, the IEEE community has developed the Next Generation Fronthaul Interface (NFGI), known as IEEE 1914 standard [23]. The NGFI interface defines options from 1 to 8, which determines the way the C-RAN (Centralized Radio Access Network) are deployed (Figure 3).

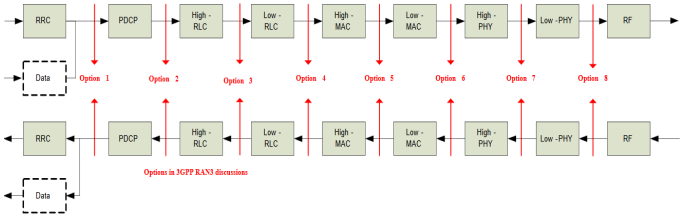

Figure 3. Functional C-RAN splits defined by the CPRI (courtesy of IEEE1914) [24]

The reason for splitting the functions is that this model will allow the future 5G massive MIMO systems to exhibit cost-effective capabilities by splitting the radio functions between BBUs (Baseband Units) equivalent to Centralized Units (CU) and RRUs (Remote Radio Units) or Remote Radio-Heads (RRH), equivalent to Distributed Units (DU) [23].

The unequivocal functional splits described in this paper are the OpenAirInterface IF4, IF4p5 and IF5 split interfaces, applying the special NFGI interface to connect to the BBU pool in the cloud edge datacenter, closer to the end user and directly communicating with RRU radio frontend. Explicitly,

the IF4 NGFI, also known as option 7, refers to the "resource mapping and IFFT (Inverse Fast-Fourier Transform)" functionalities, as well as "FFT and Resource de-mapping". The IF4p5 manages the fronthaul RX/TX precoding (NGFI RCC), whereas the IF5 is the eNB BBU functionality [25].

This way, the model is formed as C-RAN (Centralized or Cloud Radio Access Network), which is the base form of the next-generation models of 5G architecture. By splitting the architecture, various scenarios and possibilities for network slicing, service determination, scaling and performance shaping are opened (as with bandwidth proliferation, so as with latency and jitter reduction). With this achievement, a way is paved towards forming a fully functional 5G network using proprietary 4G virtualized hardware without the requirement for the NR (New Radio) 5G interfaces, and as the technologies emerge, implement the 5G new radio entities subsequently. Furthermore, it is showed that the differences in the core networking between 4G and 5G are minimal, that can result also into consecutive interworking between the two technologies and therefore, as planned, reuse the existing 4G EPC network cores and implement them together with the new 5G cores [27][28].

IV. Implementation

The functional split is the essential aspect of the new gNB (next-generation Node-B) in 5G, as it should also interwork with the 4G eNB, as well as with various other access technologies (Wi-Fi, WiMAX etc.). An intriguing part of the RRU, is that there are various techniques and interfaces which allow different devices to access the 5G network in a same manner as a typical 4G LTE access device (with same PDCP message exchange etc.) by using Wi-Fi or any other wireless access technology. The issue with sending data from the Baseband Unit to the EPC through containerized environments can be fixed by allowing direct communication between the containers and the packaged applications.

According to Figure 4, the communication between the RRU and the EPC is redirected through the BBU Docker container (green), residing in the network edge. The centralized unit utilizes the Calico BGP virtual agent [29] that can talk to the remote EPC deployed in the OpenStack cloud (red). The mechanism that allows inter-container direct link is the bypassing of layer-2 stacks created by the Open vSwitch in the OpenStack Neutron; namely, the plugin Kuryr integrates the Docker daemon with the OvS by authenticating the virtual machine with the Keystone service.

It provides access to the Neutron networking with the specific user that is assigned to (in this case Admin, within the Admin network, the same project name and relevant password). Notably, in this version of OpenStack, the actual version of Keystone is v2.0, that has slightly different API than v3.0 and uses less authentication parameters, disregarding the need for the "project_domain_name", "project_domain_id", "user_domain_name" and "user_domain_id" fields.

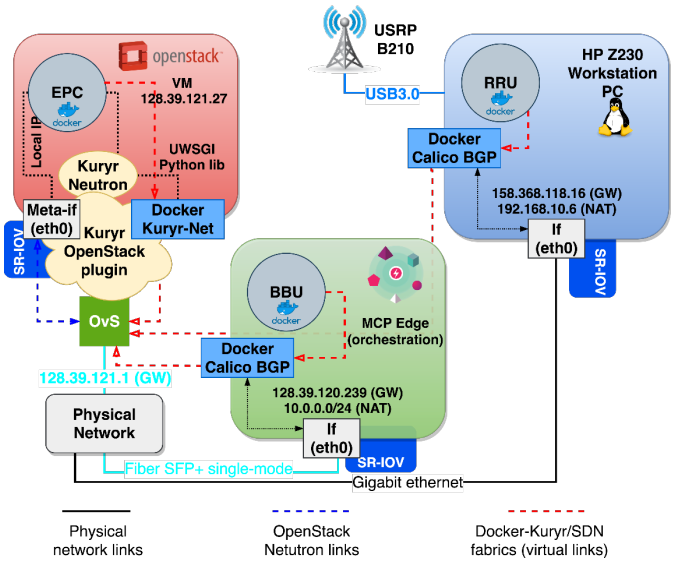

Figure 4. Established OpenStack network underlay with Kuryr plugin

As Kuryr is integrated with the Docker daemon in the VM. At this point, it has relevant control for creating and editing Neutron networks in the specified user (Figure 5) by utilizing the IPAM driver of Kuryr. With this, Kuryr enables the EPC container to communicate without the SNAT restrictions in OpenStack, utilizing full layer-3 approach.

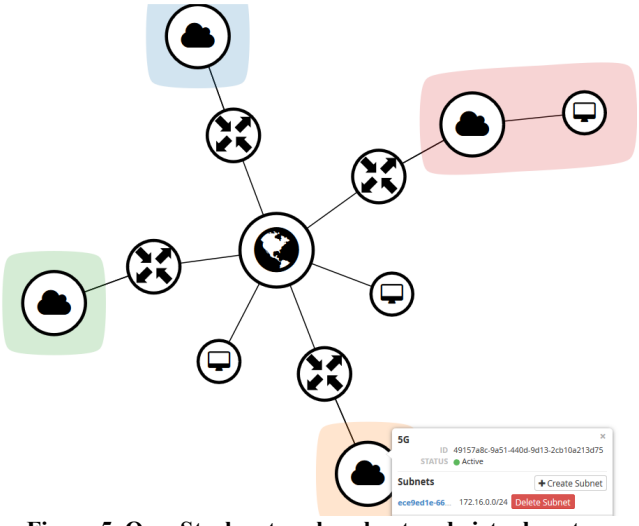

Figure 5. OpenStack network, subnet and virtual router created by the Kuryr plugin

A. *Providing Identity Federation*

The Cloud Computing paradigm (for its services and infrastructures) is also suited when considering **IoT** devices as we also see a massive adoption of such solutions by several entities to have their management platforms for such devices. As our focus goes towards the ones that are open-source, we are considering (as it was mentioned before) OpenStack [5] and, for the purpose of **IAM**, Keystone [30] is the tool that allows to manage the access to all of the components of the solution. Recently Keystone has integrated functionalities that allow using identity federation protocols such as OpenID Connect [31] from a third-party perspective, which means that it allows adding to its authentication flow third-party **IDP**s.

Gluu Server [32] offers identity management and provision services that allow establishing authentication mechanisms in network resources.

This component will be responsible for issuing the identities for every device that tries/will be registered in the network and co-managing with Keystone [30] their usage. In order to have this link between the **IDP** server and the Cloud component, it means that it is necessary to achieve an agreement and a consensus so that the services can communicate with the platform. Thus, a third-party integration with Keystone [30] so that the identities issued and provided by the Gluu Server [12][32] can be used and recognized by the platform, allowing to achieve a federation regarding the type of identity being used by all the elements existing in the network.

V. Evaluation

When adopting the method with Kuryr in OpenStack for containerized appilcations, it is essential to note that the uWSGI libraries encumber the procedure with Keystone service v2.0. The new OpenStack versions introduce Keystone v3.0, which allows Kuryr to map several additional authentication flags and allow the Kuryr libnetwork driver to successfully create a virtual network, subnetwork, and a virtual router in Neutron and embody the Docker daemon with IPAM driver. An alternative for setting up the Kuryr-libnetwork driver is a Docker Kuryr image, that can be run alongside the infrastructure and automate the deployment of the Kuryr plugin.

VI. Conclusion

The Kuryr-libnetwork driver for Docker allows the daemon to translate networking procedures directly into the instance at which the containerized application is executed. With minimal regulation, it is feasible to attain substantially granular control over the Neutron networking service in OpenStack, for the user at which the Kuryr plugin is authenticated. Consequently, we showed that the explicit requirement for tunneling between the remote eNB location and the EPC running in the cloud is negated, setting a simplified method that can be utilized across different public and private clouds.

Acknowledgment

This paper is a result of the SCOTT project (www.scott-project.eu) which has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This Joint undertaking receives support from the European Union's Horizon 2020 research and innovation program and several countries such as Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium and Norway.

References

[1] IEEE 802.11 Working Group, 2018, *http://www.ieee802.org/11/*

[2] ZigBee Standard 802.15.4 Specification, 2014, *https://www.zigbee.org/download/standards-zigbee-specification/*

[3] Bluetooth Low Energy (BLE) Core Specification, 2017, *https://www.bluetooth.com/specifications/bluetooth-core-specification*

[4] OpenAirInterface Working Group, *https://www.openairinterface.org/*

[5] OpenStack Cloud Software, https://www.openstack.org

[6] Docker, 2018, *https://www.docker.com/*

[7] Kubernetes, 2018, *https://kubernetes.io/*

[8] Neutron Networking Service for OpenStack cloud, 2018 *https://docs.openstack.org/neutron/latest/*

[9] OpenStack Kuryr Plugin, *https://wiki.openstack.org/wiki/Kuryr*

[10] RedHat – uWSGI (Web Server Gateway Interface), *https://uwsgi-docs.readthedocs.io/en/latest/*

[11] SR-IOV (Single Root – Input/Output Virtualization) Support in Kuryr, *https://docs.openstack.org/kuryr-libnetwork/latest/config-sriov.html*

[12] B. Santos, V. T. Do, B. Feng, and T. van Do, "Identity Federation for Cellular Internet of Things," in Proceedings of the 2018 7th International Conference on Software and Computer Applications - ICSCA 2018, 2018, pp. 223–228

[13] RFC1701: GRE encapsulation, *https://tools.ietf.org/html/rfc1701*

[14] RFC2547: BGP/MPLS IP Virtual Private Networks (VPNs), *https://tools.ietf.org/html/rfc4364*

[15] RFC3931: Layer Two Tunneling Protocol Version 3 (L2TPv3), *https://tools.ietf.org/html/rfc3931*

[16] RFC6071: IP Security (IPSec) and Internet Key Exchange (IKE) Document Roadmap, *https://tools.ietf.org/html/rfc6071*

[17] RFC2637: Point-to-Point Tunneilng Protocol (PPTP), *https://tools.ietf.org/html/rfc2637*

[18] Minimal Encapsulation within IP, *https://tools.ietf.org/html/rfc2004*

[19] OpenStack "Mitaka", *https://www.openstack.org/software/mitaka/*

[20] OpenStack "Pike" Security Rules, *https://docs.openstack.org/python-openstackclient/pike/cli/command-objects/security-group-rule.html*

[21] Tungsten Fabric: Mirantis Cloud Platform Software Defined Networking, *https://www.mirantis.com/software/mcp/sdn/*

[22] OpenDaylight SDN controller, *https://www.opendaylight.org/*

[23] IEEE 1914 NGFI Working Group, *http://sites.ieee.org/sagroups-1914/*

[24] T. Mustala and O. Klein, "Common Public Radio Interface (CPRI/eCPRI Overview)", IEEE LAN/MAN Standards Committee, 2017, *http://www.ieee802.org/1/files/public/docs2017/cm-mustala-eCPRI-Overview-0917.pdf*

[25] R. Knopp, N. Nikaein, C. Bonnet, F. Kaltenberger, A. Ksentini and R. Gupta, "Prototyping of Next Generation Fronthaul Interfaces (NGFI) Using OpenAirInterface", EEURECOM, France, 2018, *https://www.openairinterface.org/?page_id=1695*

[26] B. Dzogovic, V. T. Do, B. Feng and T. van Do, "Building virtualized 5G networks using open source software," *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, Penang, 2018, pp. 360-366. doi: 10.1109/ISCAIE.2018.8405499, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8405499&isnumber=8405426

[27] N. Makris, C. Zarafetas, P. Basaras, T. Korakis, N. Nikaein and L. Tassiulas, "Cloud-Based Convergence of Heterogeneous RANs in 5G Disaggregated Architectures," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, 2018, pp. 1-6. doi: 10.1109/ICC.2018.8422227, URL: *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8422227&isnumber=8422068*

[28] N. Makris, P. Basaras, T. Korakis, N. Nikaein and L. Tassiulas, "Experimental evaluation of functional splits for 5G cloud-RANs," *2017 IEEE International Conference on Communications (ICC)*, Paris, 2017, pp. 1-6, doi: 10.1109/ICC.2017.7996493, URL: *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7996493&isnumber=7996317*

[29] Calico SDN controller, 2018, *https://www.projectcalico.org*

[30] OpenStack KeyStone, *https://docs.openstack.org/keystone/*

[31] OpenID Connect, *https://openid.net/connect/*

[32] Gluu Server, *https://www.gluu.org*