Flexible 4G/5G Testbed Setup for Mobile Edge Computing using OPENAIRINTERFACE and OPEN SOURCE MANO

Thomas Dreibholz
Simula Metropolitan Centre for Digital Engineering
c/o OsloMet – storbyuniversitetet
Pilestredet 52, 0167 Oslo, Norway
dreibh@simula.no

Abstract Setting up a working 4G/5G mobile network development testbed can be a highly complicated and error-prone task. In this paper, we therefore introduce our open source Virtual Network Function (VNF) for an OPENAIRINTERFACE-based Evolved Packet Core (EPC) for deployment with the Open Source Management and Orchestration (OPEN SOURCE MANO, OSM) framework. By using our VNF as basis, it will be easily possible to create own testbeds and extend them with further functionality, particularly – but not limited to – Mobile Edge Computing (MEC) setups. In a simple proof of concept, we demonstrate a basic transport protocol performance evaluation in a deployed test network.

1 Introduction

Clearly, the fifth generation of mobile broadband networks (5G) is the next major step in the evolution of mobile communications. In the very near future, 5G networks will be the base of all kinds of advanced applications, from high-speed mobile media streaming over low-latency vehicular communication and control of *Unmanned Aerial Vehicles* (UAV) to energy-efficient *Internet of Things* (IoT) communication and reliability-critical emergency networks. Currently, a significant effort by both, industry and academic research, is made to design and develop all the components of 5G networks. However, the current *Long-Term Evolution* (LTE, 4G), which is the basis of the developing 5G standards, already has a high complexity. Even setting up a small testbed setup already requires a significant amount of software to be deployed, and expensive proprietary hardware to be purchased and installed. Therefore, the goal of the OPENAIRINTERFACE consortium is the create an open source software and hardware base to realise a full 4G setup. Based on this, development towards 5G is made.

OPENAIRINTERFACE is a large consortium, with many very active members – from industry and academia – contributing to the enhancements of its software.

Therefore, it clearly reflects the state-of-the-art in 5G development. Also, this makes it a very good foundation to develop further extensions on. Particularly, this means to combine it with extensions for *Mobile Edge Computing* (MEC) services: having an own 4G/5G mobile network testbed setup, it is possible to develop, deploy and test own MEC services for research and evaluation purposes. Since OPENAIRINTERFACE is free, open source and running on of-the-shelf hardware, it allows people to get involved into MEC development without high investments into proprietary equipment and software.

However, setting up OPENAIRINTERFACE [10] is not a simple and easy task. Even the most basic 4G/5G testbed setup requires a significant complexity of deploying, configuring and monitoring the underlying software components. Keeping track with the ongoing fast development pace of OPENAIRINTERFACE makes this even more difficult. Therefore, for simplifying and automating such testbed setups, we have developed an open source *Virtual Network Function* (VNF) for an *Evolved Packet Core* (EPC) based on OPENAIRINTERFACE. This EPC VNF can be deployed easily by OPEN SOURCE MANO (OSM) in a cloud setup (e.g. based on OPENSTACK). Using our VNF as network basis, it is therefore straightforward to plug it together with further functionalities for MEC, in order to build an advanced test network for MEC-based services and applications. In this paper, we introduce our VNF and show a basic proof-of-concept evaluation.

2 Software

Before we introduce our EPC VNF, we briefly explain the underlying software. Since EPC, OPENAIRINTERFACE and OSM make heavy usage of abbreviations, we will shortly introduce the basic ones here. This will particularly also be helpful for the reader to understand the referenced detailed documentations.

2.1 OPENAIRINTERFACE *(OAI)*

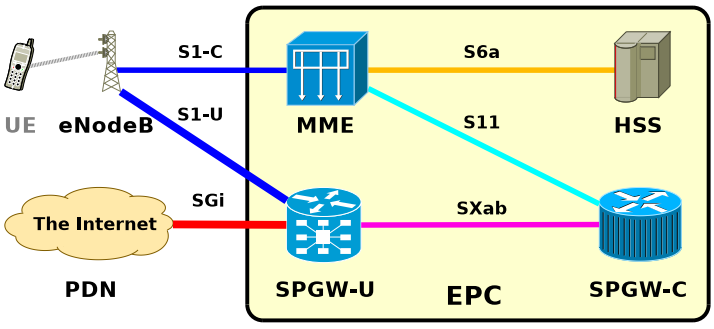


Fig. 1 The OPENAIRINTERFACE Components

OPENAIRINTERFACE[1] (OAI) is a non-profit consortium developing an open source software and hardware solution to realise the EPC, access network and user equipment of cellular networks. It provides software for the following components [10]:

HSS: The *Home Subscriber Server* (HSS) is the central database containing the information about users and their subscriptions. The HSS functionalities include mobility management, session establishment, user authentication and access authorisation. It provides its service to the MME via the S6a interface.

MME: The *Mobility Management Entity* (MME) handles the procedures of attaching and detaching as well as service requests of *User Equipment* (UE) and eNodeBs. It communicates with eNodeBs over the S1-C interface, with SPGW-C over the S11 interface, and with HSS over the S6a interface.

SPGW-C: The *Control Plane of the Packet Data Network Gateway* (SPGW-C) provides the control part of a combined Serving Gateway (SGW) and Packet Data Network Gateway (PGW). That is, OAI combines SGW and PGW, but uses *Control and User Plane Separation* (CUPS). The SPGW-C handles control requests from the MME via the S11 interface, and communication with the SPGW-U via the SXab interface.

SPGW-U: The *User Plane of the Packet Data Network Gateway* (SPGW-U) handles the forwarding of user traffic between the *Public Data Network* (PDN) at the SGi interface (i.e. usually the public Internet) and the eNodeB over the S1-U interface. User traffic between eNodeB and SPGW-U is tunnelled via GPRS Tunnelling Protocol (GTP). The setup of user traffic tunnels is controlled by the SPGW-C over the SXab interface.

eNodeB: The *Evolved Node B* (eNodeB) realises a base station. For this purpose, the eNodeB implementation of OAI supports a couple of *Software-Defined Radio* (SDR) hardwares (e.g. ETTUS USRP B210), connected to a computer via USB 3.x port. Although the computation-intensive part is mostly done by the *Field-Programmable Gate Array* (FPGA) in hardware, the software part – running on a standard Linux PC – is timing-critical. It therefore requires a low-latency real-time kernel[2]. It is therefore strongly recommended to run the eNodeB on a dedicated PC.

The OAI components can be set up on one or more PCs and/or virtual machines. However, OAI provides no management and orchestration software.

2.2 OPEN SOURCE MANO *(OSM)*

OPEN SOURCE MANO[3] (OSM) is an open source *Management and Orchestration* (MANO) framework [14]. As an operator-led community, OSM is offering a

[1] OPENAIRINTERFACE: https://www.openairinterface.org.

[2] Linux kernel compiled with the option CONFIG_PREEMPT_RT.

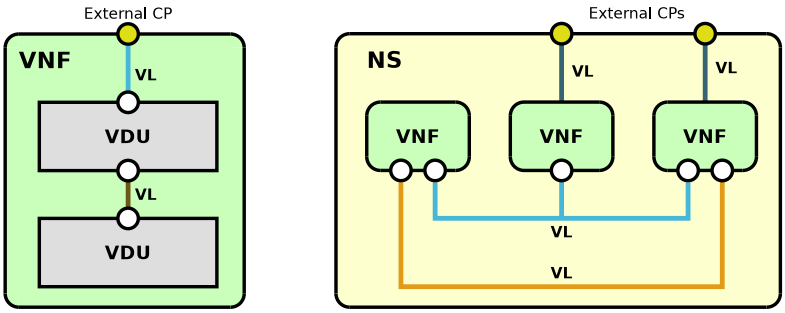
[3] OPEN SOURCE MANO: https://osm.etsi.org.

production-quality open source MANO stack that meets the requirements of commercial Network Function Virtualisation (NFV) networks. NFV with OSM can be split into three parts [14, Chapter 1]:

NFV Infrastructure (NFVI): NFVI denotes the part hosting virtual machines and containers, as well as connecting them by *Virtual Links* (VLs). OSM is independent of the underlying hosting mechanism. In many cases, hosting is realised by OPENSTACK [11, 13], but other solutions like VMWARE are supported as well.

VNFs, NSs and Network Slices: The second part is the collection of VNFs and their interconnection and composition into *Network Services* (NS). NS can further be composed and shared to form network slices.

MANO: The last part is the management and orchestration system. It controls the life-cycle of VNFs, NSs and network slices, including their configuration and monitoring.



(a) VNFD (b) NSD

Fig. 2 Virtual Network Function Descriptor (VNFD) and Network Service Descriptor (NSD)

Realising and deploying a VNF as part of a NS is introduced in detail in [8]. We therefore only briefly summarise it here:

VNFD: A VNF is created in form of a *VNF Descriptor* (VNFD). The VNFD contains a definition of its *Virtual Deployment Units* (VDU), which correspond to an own *Virtual Machine* (VM). Subfigure 2(a) illustrates an example: a VNFD defines a VNF consisting of two VDUs. *Connection Points* (CP) define interfaces (CPs of a VDU appear as network interfaces in a VM instance). CPs can be connected by *Virtual Links* (VL). VDUs of the VNF are connected by using internal CPs. In the example, the two VDUs are connected via a VL. This VL is referenced by its VL Descriptor (VLD), which can be a name like e.g. "s6a". CPs of VDUs can also be connected to external CPs of the VNF. In the example shown by Subfigure 2(a), the upper VDU has such a connection.

NSD: A *NS Descriptor* (NSD) connects VNFs with VLs. Furthermore, it has the possibility to also define external NS CPs, e.g. for attaching them to physical networks in the underlying NFVI. Subfigure 2(b) presents an example consisting of three VNFs with interconnecting VLs and two external CPs. Note, that a VL can connect multiple CPs (i.e. a "virtual switch", not just a "virtual cable").

Instantiating an NSD in OSM, creating a new NS, triggers the setup of VMs in the underlying NFVI (e.g. in OPENSTACK). Furthermore, the interconnecting virtual links are realised by creating virtual networks, switches, links, etc.. This is denoted as "day-0 configuration". The VM instances run the operating system image defined as part of the corresponding VDU (e.g. a specific "UBUNTU 18.04" image). In most cases, however, the image does not contain the full installation for the VDU. Therefore, the VDU definition as part of the VNFD allows customised configuration by using a JUJU[4] Charm. A JUJU Charm is a container running on the OSM machine. It maintains an SSH connection to its VM in order to configure and monitor it. This SSH connection is made over a designated management network (i.e. the VNFDs and NSD need corresponding CP and VL definitions). Particularly, a Charm allows executing commands on the VM, e.g. to install and configure software ("day-1 configuration"), as well as to later maintain and update the setup ("day-2 configuration").

3 The SimulaMet OAI EPC

In the following, we describe our EPC VNF, denoted as SimulaMet OAI EPC, its configuration with JUJU, and its usage as part of an NS.

3.1 Virtual Network Function

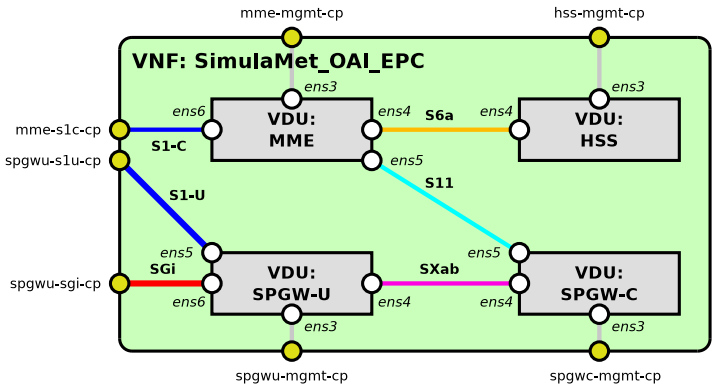


Fig. 3 The SimulaMet EPC VNFD

The VNFD of the SimulaMet OAI EPC VNF is illustrated in Figure 3. Clearly, it consists of a VDU for each of the four OAI EPC parts introduced in Subsection 2.1 and Figure 1: HSS, MME, SPGW-C and SPGW-U. The interfaces S6a, S11 and SXab are realised by VLs. S1-C, S1-U and SGI are attached to external CPs. In

[4] JUJU: https://jaas.ai.

an NSD, they can be attached to e.g. physical interfaces or VLs connecting them to other VNFs (e.g. to add MEC components). Our VNF uses an UBUNTU 18.04 "Bionic Beaver" operating system image; the interface names `ens3`, ..., `ens6` are the interface names in the virtualised Linux systems.

3.2 Configuration with JUJU *Charms*

Table 1 Basic Parameters for the SimulaMet EPC VNF

Parameter	Description	Example
network_realm	Network realm	simula.nornet
network_op	Network Operator Code OP	1006020f0a478bf6b699f15c062e42b3
network_k	Network Subscriber Key K	449c4b91aeacd0ace182cf3a5a72bfa1
network_mcc	Network MCC	242
network_mnc	Network MNC	88
network_imsi_first	First IMSI	242881234500000
network_msisdn_first	First MSISDN	24288880000000
network_users	Number of users	1024
network_ipv4_dns1	Primary DNS address	8.8.8.8
network_ipv4_dns2	Secondary DNS address	8.8.4.4
hss_git_repository	HSS GIT repository	https://github.com/OPENAIRINTERFACE/openair-cn.git
hss_git_commit	HSS GIT commit	2019.w45
hss_S6a_address	HSS S6a address	172.16.6.129
mme_git_repository	MME GIT repository	https://github.com/OPENAIRINTERFACE/openair-cn.git
mme_git_commit	MME GIT commit	2019.w45
mme_S1C_ipv4_interface	MME S1-C interface	192.168.247.102/24
mme_S1C_ipv4_gateway	MME S1-C gateway	0.0.0.0
mme_S11_ipv4_interface	MME S11 interface	172.16.1.102/24
mme_S6a_address	MME S6a address	172.16.6.2
spgwc_git_repository	SPGW-C GIT repository	https://github.com/OPENAIRINTERFACE/openair-cn-cups.git
spgwc_git_commit	SPGW-C GIT commit	2019.w47
spgwc_S11_ipv4_interface	SPGW-C S11 interface	172.16.1.104/24
spgwu_git_repository	SPGW-U GIT repository	https://github.com/OPENAIRINTERFACE/openair-cn-cups.git
spgwu_git_commit	SPGW-U GIT commit	2019.w47
spgwu_S1U_ipv4_interface	SPGW-C S1-U interface	192.168.248.159/24
spgwu_S1U_ipv4_gateway	SPGW-C S1-U gateway	0.0.0.0
spgwu_SGi_ipv4_interface	SPGW-C SGi interface	10.254.1.203/24
spgwu_SGi_ipv4_gateway	SPGW-C SGi gateway	10.254.1.1

Instantiating our VNF requires to define an NSD (in the simplest case: just consisting of the VNF and attaching it to physical interfaces and a management network for the JUJU Charms) and provide the parameters listed in Table 1. In particular, these parameters provide the information about the mobile network to be created, i.e. realm, *Mobile Country Code* (MCC), *Mobile Network Code* (MNC), *Operator Code* (OP), *Subscriber Key* (K), as well as the first user's *International Mobile Subscriber Identity* (IMSI) and *Mobile Station International Subscriber Directory Number* (MSISDN). IMSI and MSISDN are incremented for further users. The JUJU Charms of the VNF will create the given number of users and subscrip-

tions according to these details; simcards for the new mobile network have to match these settings. Further parameters include the addressing of the components and the addresses of the *Domain Name Servers* (DNS) to be used.

OAI is under heavy development, and working with branches and forks of the OAI sources is an essential part of development and tests with OAI. Therefore, the JUJU Charms of our VNF not just deploy pre-built variants of HSS, MME, SPGW-C and SPGW-U. Instead, the parametrisation allows to specify the corresponding GIT repository and tag for each of the four components. During day-1 configuration, each component is therefore directly compiled from its sources inside the corresponding VDU. For this purpose, the operating system image already has development packages, e.g. compilers, libraries, etc., preinstalled. A script is provided to create this image from scratch, by downloading and automatically installing a preseeded UBUNTU Linux. Since the VDUs need Internet access for the build (e.g. to fetch the sources and dependencies), it is necessary to allow Internet access over the management network.

While the management network provides Internet access for package and source downloads during day-1 and day-2 configuration, it is not desirable that the PDN access of the mobile network goes over the management network as well. Therefore, the SPGW-U Charm also deploys a routing rule setup for the SPGW-U VDU: the OAI SPGW-U configures a dummy device `pdn`. Network/port address translation is then used for forwarding all PDN traffic via the public IPv4 address of the SGi interface `ens6`. Our SPGW-U Charm configures a new routing rule for all traffic from `pdn`: all user traffic is routed via the SGi interface `ens6` to the PDN, without routes for the internal networks. So, UE is only able to access the SGi network and the PDN, but unable to interfere with the internal networks.

As last step of the day-1 configuration, the JUJU Charm of each component sets up a SYSTEMD unit configuration for the corresponding service. This service is then automatically started, i.e. once the NS is fully configured, all components should be running, and the new mobile network is ready for attaching eNodeBs.

The SimulaMet OAI EPC VNFD sources, together with example NSDs, build tool-chain and test scripts, operating system image build script as well as documentation is available as open source in form of a public GIT repository under https://github.com/simula/5gvinni-oai-ns.

4 Testbed Setup

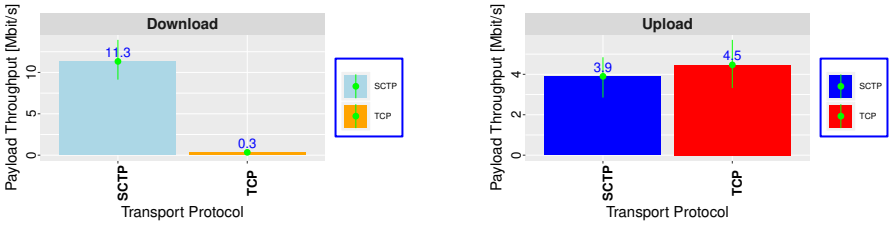
For our proof-of-concept evaluation, we deployed an OPENSTACK setup [12] based on OPENSTACK "Stein" on UBUNTU 19.04 "Disco Dingo". A management network with Internet access was configured, in addition to a further network with Internet access for the PDN. Furthermore, we deployed the latest OSM version "Release SEVEN" on UBUNTU 18.04 "Bionic Beaver". Another dedicated UBUNTU 18.04

server running in a network of our site provided a peer instance for the NETPERFMETER[5] [1, Section 6.3] [2,4] transport protocol performance evaluation tool.

Our VNF was then instantiated by OSM into OPENSTACK, with the SGi interface configured into our OPENSTACK PDN network, as well as S1-U and S1-C connected to the eNodeB networks. The VNF instance used the "2019.w45" tags of the HSS and MME sources[6], as well as the "2019.w47" tags of the SPGW-U and SPGW-C sources[7]. Furthermore, we deployed an OAI eNodeB using tag "2019.w44" of the eNodeB sources[8] on a dedicated UBUNTU 18.04 "Bionic Beaver" laptop with ETTUS USRP B210 SDR board attached over USB 3.0. As UE, we used an UBUNTU 20.04 "Focal Fossa" laptop with HUAWEI E392 USB modem.

For our experiment, we examined three different scenarios: (1) download from server to UE, (2) upload from UE to server, and (3) bidirectional communication between UE and server. We tested TCP and SCTP [1, Chapter 3] as transport protocols. For SCTP, we used messages of 1400 bytes (i.e. resulting in one packet per message without need for segmentation). The parameters for send and receiver buffer sizes, as well as the congestion control had been left at the Linux kernels' default settings. Explicit Congestion Notification (ECN) [6] support had been enabled. The duration of each transmission run was 20 s. All runs had been repeated at least 10 times; the results show average values together with their 95% confidence intervals.

5 Results



(a) "Download": Reception Rate at UE (b) "Upload": Reception Rate at Server

Fig. 4 Application Payload Throughput for Reception in the "Download" and "Upload" Scenarios

Figure 4 presents the application payload reception rate results for the "Download" scenario (at the UE, Subfigure 4(a)) and the "Upload" scenario (at the server, Subfigure 4(b)) for both transport protocols in Mbit/s. For using SCTP, the results are as expected in both directions, with around 11.3 Mbit/s for download to the UE, and 3.9 Mbit/s for upload to the server.

For TCP, we would have expected slightly higher results, since the mobile network *should* not interfere with higher-level protocols. However, the download speed

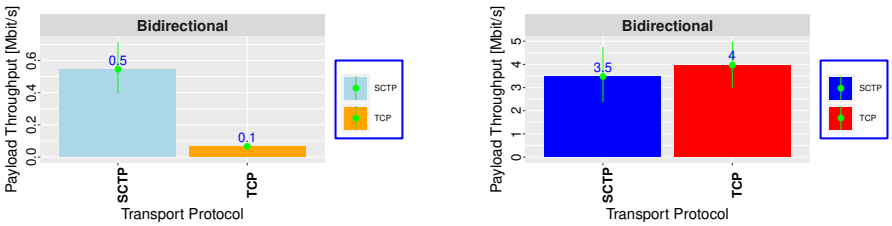
[5] NETPERFMETER: https://www.uni-due.de/~be0001/netperfmeter/.

[6] OAI HSS and MME: https://github.com/OPENAIRINTERFACE/openair-cn.git.

[7] OAI SPGW-U and SPGW-C: https://github.com/OPENAIRINTERFACE/openair-cn-cups.git.

[8] OAI eNodeB: https://gitlab.eurecom.fr/oai/openairinterface5g.git.

with TCP just reaches around 0.3 Mbit/s, while the upload speed reaches 4.5 Mbit/s. Since the overhead for TCP is smaller than for SCTP (TCP can utilise the full MTU size, while SCTP messages of 1400 bytes generate one packet per message), the higher TCP upload rate is as expected. As part of ongoing work, we are currently investigating the reason for this significant download speed difference, in order to identify performance problems and/or bugs in OAI.



(a) "Bidirectional": Reception Rate at UE (b) "Bidirectional": Reception Rate at Serv.

Fig. 5 Application Payload Throughput for Reception in the "Bidirectional" Scenario

The results for the "Bidirectional" scenario are presented in Figure 5, with the application payload reception rate at the UE in Subfigure 5(a) and at the server in Subfigure 5(b). As expected for the server side, the SCTP reception rate is slightly lower than in the unidirectional "Upload" scenario: SCTP's selective acknowledgements [1, Chapter 3] for received chunks cause additional overhead. The reception rate at the UE side is significantly lower than for the unidirectional "Download" scenario for both, SCTP (ca. 0.5 Mbit/s vs. 11.3 Mbit/s) and TCP (ca. 0.1 Mbit/s vs. 0.3 Mbit/s). Particularly the performance drop for SCTP cannot be explained by additional overhead for the acknowledgements. As part of ongoing work, it is therefore necessary to further investigate the OAI software, in order to identify and solve performance problems and/or bugs.

In summary, we have shown that our setup is working, with some likely bugs and performance issues in the experimental OAI software used. However, since the goal of our VNF is to easily instantiate mobile network testbeds with different versions and variants of the underlying software, our system can be very helpful in supporting the OAI developers to identify problems as well as to evaluate bugfixes and enhancements, in order to provide the users an easy-to-use deployment base for their ongoing research and development work on 5G systems.

6 Conclusions and Future Work

Setting up a mobile network development testbed can be a highly complicated and error-prone task. In this paper, we have therefore introduced our open source VNF for an OPENAIRINTERFACE-based EPC for deployment with OPEN SOURCE MANO (OSM). By using our VNF as basis, it will be easily possible to create own

testbeds and extend them with further functionality, particularly – but not limited to – MEC setups. In a simple proof of concept, we demonstrated a basic transport protocol performance evaluation in a deployed test network.

As part of future work, we are going to extend our VNF by more advanced features of OPENAIRINTERFACE, particularly with support for network slicing and scaling. Furthermore, we intend to add some example MEC NSs, like e.g. the video streaming service proposed in [9]. Furthermore, we would also like to extend the underlying OSM NFV infrastructure support with multi-cloud features [5, 7], i.e. to utilise different cloud setups, with support of the MELODIC multi-cloud framework [3].

References

1. Dreibholz, T.: Evaluation and Optimisation of Multi-Path Transport using the Stream Control Transmission Protocol. Habilitation treatise, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems (2012)
2. Dreibholz, T.: NetPerfMeter: A Network Performance Metering Tool. Multipath TCP Blog (2015)
3. Dreibholz, T.: MELODIC at Hainan University: An Introduction to the MELODIC Project. Keynote Talk at Hainan University, College of Information Science and Technology (CIST) (2019)
4. Dreibholz, T., Becke, M., Adhari, H., Rathgeb, E.P.: Evaluation of A New Multipath Congestion Control Scheme using the NetPerfMeter Tool-Chain. In: Proceedings of the 19th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–6. Hvar, Dalmacija/Croatia (2011). ISBN 978-953-290-027-9
5. Dreibholz, T., Mazumdar, S., Zahid, F., Taherkordi, A., Gran, E.G.: Mobile Edge as Part of the Multi-Cloud Ecosystem: A Performance Study. In: Proceedings of the 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 59–66. Pavia, Lombardia/Italy (2019). DOI 10.1109/EMPDP.2019.8671599. ISBN 978-1-7281-1644-0
6. Fairhurst, G., Welzl, M.: The Benefits of using Explicit Congestion Notification (ECN). Internet Draft draft-ietf-aqm-ecn-benefits-08, IETF (2015)
7. Hong, J., Dreibholz, T., Schenkel, J.A., Hu, J.A.: An Overview of Multi-Cloud Computing. In: Proceedings of the 1st International Workshop on Recent Advances for Multi-Clouds and Mobile Edge Computing (M2EC) in conjunction with the 33rd International Conference on Advanced Information Networking and Applications (AINA), pp. 1055–1068. Matsue, Shimane/Japan (2019). DOI 10.1007/978-3-030-15035-8_103. ISBN 978-3-030-15034-1
8. Lavado, G.: OSM VNF Onboarding Guidelines. White paper, ETSI (2019)
9. Luo, Y., Zhou, X., Dreibholz, T., Kuang, H.: A Real-Time Video Streaming System over IPv6+MPTCP Technology. In: Proceedings of the 1st International Workshop on Recent Advances for Multi-Clouds and Mobile Edge Computing (M2EC) in conjunction with the 33rd International Conference on Advanced Information Networking and Applications (AINA), pp. 1007–1019. Matsue, Shimane/Japan (2019). DOI 10.1007/978-3-030-15035-8_99. ISBN 978-3-030-15034-1
10. OpenAirInterface: OpenAirInterface Software Installation Support (2019)
11. OpenStack: OpenStack Architecture Design Guide (2019)
12. OpenStack: OpenStack Installation Guide (2019)
13. OpenStack: OpenStack Operations Guide (2019)
14. Reid, A., González, A., Armengol, A.E., de Blas, G.G., Xie, M., Grønsund, P., Willis, P., Eardley, P., Salguero, F.J.R.: OSM Scope, Functionality, Operation and Integration Guidelines. White paper, ETSI (2019)