Open-source RANs in practice: an over-the-air deployment for 5G MEC

Juuso Haavisto, Muhammad Arif, Lauri Lovén, Teemu Leppänen and Jukka Riekki
University of Oulu, Oulu, Finland
{firstname.lastname}@oulu.fi

***Abstract*—Edge computing that leverages cloud resources to the proximity of user devices is seen as the future infrastructure for distributed applications. However, developing and deploying edge applications, that rely on cellular networks, is burdensome. Such network infrastructures are often based on proprietary components, each with unique programming abstractions and interfaces. To facilitate straightforward deployment of edge applications, we introduce open-source software (OSS) based radio access network (RAN) on over-the-air (OTA) commercial spectrum with Development Operations (DevOps) capabilities. OSS allows software modifications and integrations of the system components, e.g., Evolved Packet Core (EPC) and edge hosts running applications, required for new data pipelines and optimizations not addressed in standardization. Such an OSS infrastructure enables further research and prototyping of novel end-user applications in an environment familiar to software engineers without telecommunications background. We evaluated the presented infrastructure with end-to-end (E2E) OTA testing, resulting in 7.5MB/s throughput and latency of 21ms, which shows that the presented infrastructure provides low latency for edge applications.**

I. Introduction

Edge computing is a prominent part of current and future cellular networks. It provides cloud resources, e.g., application-specific virtualization of computation and data, in the proximity of end-users. Virtualization at the edge decouples software from hardware, e.g., based on containers, that enables a variety of software-based deployment and coordination scenarios atop common hardware. As a coherent system, this promises application performance with more bandwidth and reduced latency for user equipments (UEs) accessing the edge resources, while improving controls on privacy.

The ETSI multi-access edge computing (MEC) standards aim to deliver these promises for the local RAN atop Fifth Generation (5G) networks, allowing effective use of the up-and-coming 5G radio features. At the same time, software defined network (SDN) and network function virtualization (NFV) coupled RANs start to resemble edge platforms, as virtualization facilitates mobile network operators (MNOs) building RAN features on general-purpose hardware. This can address several scaling problems of RANs [1], including dynamic network resource allocation, and network hardware ossification.

This is the accepted version of the work. The final version will be published at the European Conference on Networks and Communications (EuCNC2019), June 18-21, Valencia, Spain, 2019.

However, edge application development and deployment are currently challenging in RANs. First, RANs, with or without MEC, have challenging requirements on E2E latency, energy consumption and reliability with both up and downlinks [2], [3]. Second, edge applications are envisioned as both data- and computation-intensive. Thus, application deployment is not straightforward concerning the capabilities of edge hosts, application requirements, and environmental parameters of the RAN. Only the consideration of all these aspects makes envisioned zero-latency applications possible. These issues are aggregated with multiple parallel applications on the same infrastructure.

Further, the current MEC standardization offers less application programming interface (API) endpoints for software development than existing cloud computing environments. Therefore, linking development and actual deployment may be hard for end-user application developers, more familiar with current cloud environments. Coincidentally, MNOs may be unable to introduce the required APIs for developers, for the RAN components being proprietary. However, if the RAN would be based on OSS, then MNO could create the required bridging interfaces. Using existing mature OSS development platforms, which support the current mainstream software engineering processes, the challenges mentioned above could be thus addressed. In this context, we present an OSS infrastructure for software-based 5G RAN development, where DevOps simplifies the deployment of E2E OTA applications to the edge, that also satisfies the attributes of a micro-operator, as discussed in [4].

The paper is organized as follows. Section II discusses the related paradigms in software and telecommunications research. Section III details the infrastructure design. Section IV evaluates the proposed infrastructure to deploy the network OTA with RAN hardware and spectrum licenses provided by the University of Oulu. Discussion follows in Section V, and lastly the contribution is concluded in Section VI.

II. Background

IoT and edge computing necessitate application development for a distributed environment, where virtualization provides a deployment mechanism. In 5G, virtualization is realized through SDN and NFV technologies. Combined, these technologies pave the way for the microservice architectural model [5], which can with DevOps practices speed up software development for the edge.

A. Virtualization

SDN and NFV use virtualization to decouple application control and data into separate planes and to provide programmability and flexibility to network management through software. This approach supports adjusting the edge infrastructure and the network to user demands while considering the locally available resources.

To realize virtualization on general-purpose hardware, containers have been reported to produce less operational overhead and better performance [6], [7] than virtual machines that are based on virtualized operating system layer instead of hardware layer [8].

From a developer's perspective, managing containers means managing applications rather than machines. This enables the same deployment environment reducing inconsistencies between development and production, which improves deployment reliability and speed.

For MNOs, containers provide an isolation mechanism which avoids edge applications from interfering with the host computer's, or other edge applications' execution. As the host computers solely manage containers, limiting operating system versions on edge hosts is easier, making the infrastructure more maintainable [9]. Container deployments spanning many physical host computers can then be managed with an orchestrator, which automates container creation, removal, restoration, and updating without interfering with the availability of the RAN or the end-user edge applications.

B. Microservices and -operators

Microservices offer a distributed, cloud-native architectural model, which refines and simplifies service-oriented architecture [5], [6]. For example, microservices follow the share-nothing philosophy, to support agile business methods and to promote software isolation and autonomy [10]. Microservices may be based on various programming languages, middleware and data stores. Each service runs in isolation while communicating through lightweight APIs. The microservice model is seen to provide isolation and scalability elastically and platform-agnostically. However, stable functional decomposition of the application into services is required with supporting system services, e.g., for service discovery, system configuration, load balancing, and fault detection.

In general, microservice development requires knowledge of distributed systems deployment, e.g., the tradeoffs between communication and computation overheads to realize the expected benefits [5], [11]. This is further challenging with MEC, which generally aims to optimize the connection between UEs and MEC services.

From the MNOs perspective, predicting the required computational and communication capabilities is challenging for a single site of operation in a traditional RAN. A distributed MEC platform adds extra variables into the challenge, such as application-specific resource migration within and between sites. Therefore, facilities for testing, deploying, and monitoring microservices becomes crucial in a cellular network due to multiple sites with varying edge computing capabilities.

In dense cellular deployments, MNOs also face challenges in allocating radio resources with limited spectrum, managing interference, and in load-balancing cells. Local geographical MNOs may thus be needed to manage wireless connectivity for UEs effectively while maximizing throughput, latency, connectivity, or any other goal [12], [13], [14]. Further, MNOs' infrastructure is challenged by ever-increasing throughput and latency requirements of the 5G applications. If resources on wider area networks need to be accessed, latencies will grow, thus on-site data centers are a viable choice for edge applications on 5G. As such, micro-operators could be the first MNOs relying solely on SDN and NFV principles, due to lack of legacy hardware and legacy application requirements from previous generations of cellular connectivity. Combined with agile business perspectives of microservices, such as the ability to cut long-term commitment and bottlenecks by realizing software as a set of complete, cohesive, and coupled services, micro-operators could thus provide an edge platform with customizability, low latency, and high reliability, that is unrealistic for traditional MNOs.

C. DevOps

DevOps paradigm provides good software quality and frequent delivery mechanisms, for development teams of any size, facilitating enhanced value creation for end-users [15]. DevOps allows software platforms to be offered as a Platform-as-a-Service (PaaS) for E2E application development. MEC applications could be independently developed and tested without the intervention of the MNO.

DevOps utilizes continuous deployment (CD), continuous integration (CI), and continuous monitoring (CM). CD and CI enable on-demand deployment of software through automated mechanisms [6]. When the number of deployments increases, as with microservices, CD and CI become essential for frequent software releases. CM then provides microservice developers and the MNO with performance-related feedback of the releases, e.g., to detect operational anomalies, and profiling.

D. Related work

Virtualized cellular networks without OTA testing have been studied widely [7], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]. OTA testbeds without MEC have been installed on LTE [26], [27], [28], [29] and, to avoid regulated spectrum, on WiFi [30], [31]. Proprietary OTA testbeds with MEC exist [32], [33], [34], [35], but without consideration of DevOps practices. In general, DevOps practices have rarely been installed on Long Term Evolution (LTE), but one study did provide Infrastructure-as-a-Service (IaaS) services [36] making such deployments possible, and another one has offered DevOps [37] on WiFi. As such, our study essentially combines the OTA environment of [36], but replaces IaaS as in [37] with PaaS, thus relieving the developer to only maintain an application instead of a server, as described in §II-A. To the best of our knowledge the research in this niche is yet uncharted.

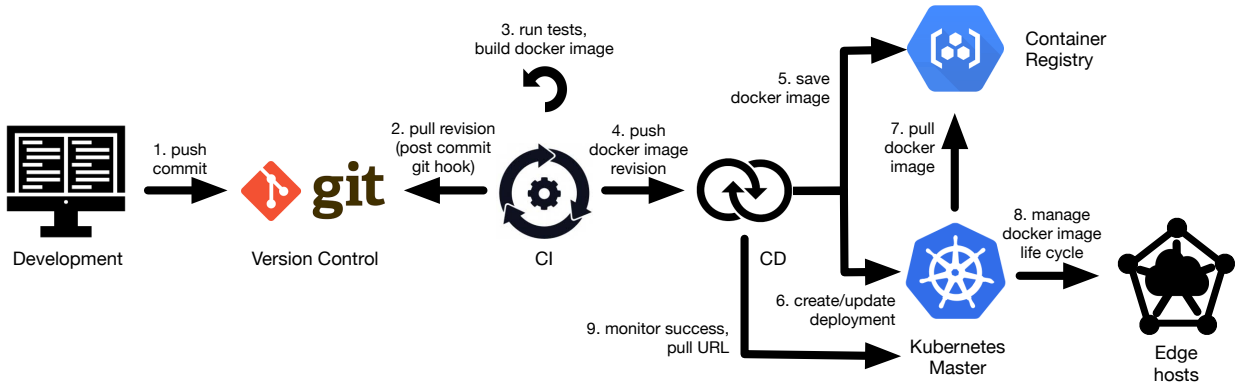

Fig. 1: Workflow and components of the DevOps platform.

III. Infrastructure Design

Testbeds are a requisite for experimental research as developed new technologies, methods, and features can be validated on-site. On the one hand, OSS testbeds are preferred for flexibility and openness, but they tend to be inferior in performance and may lack some capabilities in comparison with their proprietary counterparts. On the other hand, the business goals of commercial testbeds may differ from research goals and focus on, e.g., standardization or off-the-shelf components. As an example, current off-the-shelf software-defined radio (SDR) hardware allows building cellular deployments on virtualization and OSS [27], but the performance and features are underdeveloped.

In the corporate market, Kubernetes has emerged as the leading solution for development deployments for the edge [38], [9]. Moreover, microservice infrastructures have been promoted by major cloud computing infrastructure providers, such as Google, Microsoft, and Alibaba, with varying capabilities for management.

As we set the requirement to use the same infrastructure in both development and deployment, we built our platform with Kubernetes. In this regard, Kubernetes provides built-in isolation, prioritization, and scheduling mechanisms for separating developer applications from the actual deployments on the same infrastructure. Kubernetes supports Docker for container-based application deployment, which we used for microservice development in the proposed infrastructure. Docker provides lightweight virtualization for distributed edge application components; thus widespread deployments become possible with minimized hardware requirements.

Fig. 1 presents an overview of the proposed workflow, where DevOps services are installed on MEC hosts on Kubernetes. Thanks to industry adoption of Kubernetes in software engineering, open-source DevOps services like revision control (e.g., Gitea), CI and CD (e.g., Drone), and CM (e.g., Prometheus) are easy to integrate. By installing these DevOps tools on Kubernetes we base the initial configuration of the testbed on OSS, avoiding dependency on a single tool provider. Should a DevOps component prove to be unsuitable for our needs, we can replace it more cost-effectively than a proprietary cloud-based service. As a result, the unity of these components contains most of the functionality specified in the MEC reference architecture [39] (Fig. 6-1). The reference architecture is further described in [40] (§V.B), with which we can translate the ETSI standard to Kubernetes terminology: APP is a deployment in Kubernetes, VIM is a container registry, CFS is kubectl interfacing through Web UI, and LCM is kubectl interfacing through JSON API.

New E2E edge applications are deployed as follows (see Fig. 1): a developer places application source-code in a git-repository. Once the developer pushes changes to the master branch of the repository (1), an automatic post-commit git hook pulls the source-code to the CI server (2), which runs the developer-specified tests (3), and if successful, attempts to build the application using its Dockerfile. If successful, the CI creates a Docker image, passes it to CD (4), which saves it (5), and commands Kubernetes to deploy the application using the built Docker image (6, 7). If Kubernetes is able to provision the application (8), the CD server returns the application's DNS address from Kubernetes (9), which is then passed onto the developer to be used to access the application from an UE.

Fig. 2 shows the software stacks of the primary host running Kubernetes master and the EPC, and the edge hosts meant solely for hosting the edge applications. The primary host is built on Arch Linux, as it provides straightforward installation of a real-time Linux kernel. The edge hosts operate on CoreOS, that runs containers natively, and be used to form a cluster of bare-metal servers. The clusterization allows the CoreOS hosts to update the container runtime and kernel components automatically and in succession, resulting in zero-downtime operating system upgrades [41]. Each physical host then runs the standard Kubernetes architecture, e.g., flannel as a CNI (Container Network Interface) for establishing a virtual network that attaches IPs for containers, Kube Proxy for routing between pods, and Kubelet as orchestration agent responsible for the control interface.

A. Network architecture

Fig. 3 shows the network architecture of our infrastructure. UEs are assigned an IP from the `tun/tap` device of the EPC.

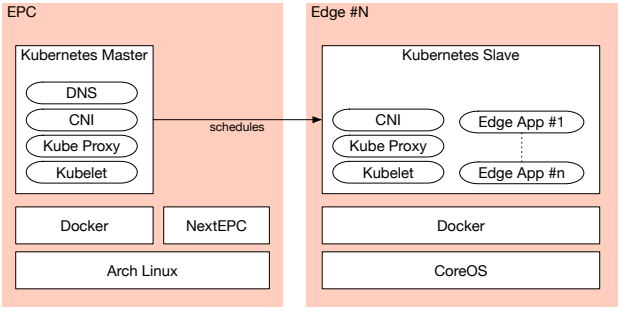

Fig. 2: Software stacks of the edge hosts. Left: the primary host running Kubernetes master and the EPC. Right: the edge hosts hosting the edge applications.

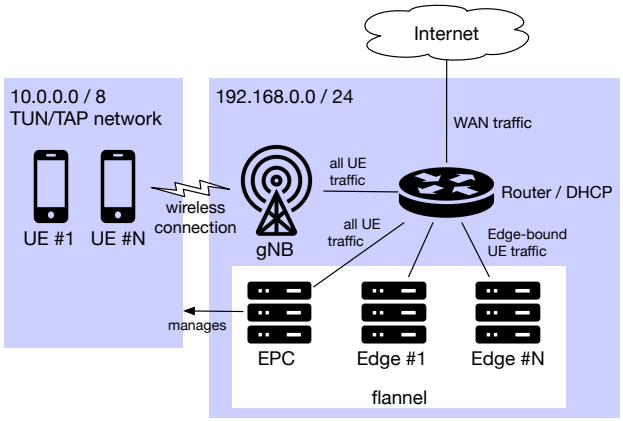

Fig. 3: Platform network architecture.

As a part of the Kubernetes cluster, the EPC can route UEs to access the rest of the Kubernetes deployment, on which the load-balanced edge applications can run in a single network hop from the EPC, facilitating guarantees on low latency.

The EPC implementation must support low latency edge applications on the RANs. Thus a real-time Linux kernel is a requirement to optimize its network stack [7]. As discussed in [42], EPC should be left unvirtualized for NFV management. Further, we installed kernel passthrough for the network interface controller (NIC) [7], [42]. As such, the EPC is a scalable software component instead of a hard to upgrade proprietary digital signal processor (DSP) hardware.

In this respect, we reviewed the current OSS EPC projects, as shown in Table I. We chose NextEPC as our environment for its ease of installation and support for 3rd Generation Partnership Project (3GPP) Release 13. Besides having the most up-to-date 3GPP conformance, NextEPC is also the only EPC installable from a package manager and with an easy-to-use web-based subscriber management interface.

TABLE I: Currently available OSS EPC packages.

Project	Language	Anecdotal summary
NextEPC	C	3GPP Release 13 support
corenet	Python	Minimal 3G and LTE EPC
vEPC	C++	For performance testing [43]
openair-cn	C	3GPP Release 10 support [44]
srsLTE	C++	3GPP Release 8 support
openLTE	C++	Elegant, but incomplete [45]

IV. Evaluation and Results

We evaluated the proposed infrastructure with regard to communication latency between one UE and the EPC as infrastructure end-point, that is the focus of the edge paradigm. The evaluations are conducted OTA using Nokia picocell as a base station, set at 10Mhz on band 7, using spectrum licenses of the University of Oulu. The EPC is installed on commodity hardware running i3-6100. The results establish a baseline on which to set the performance expectations with application-specific virtualization.

We measured latencies in four data sets, as shown in Table II and Fig. 4. The first set served an empty web page. The rest of the tests served pre-generated binary blogs generated by `/dev/rand`. The second test served a single 1MB binary file. The third test served a single 10MB binary file, and the fourth served 10MB binary file over WiFi as a baseline. All the tests sent 250 requests over 5 seconds in chunks of 50 requests per second. We used HTTP-based Vegeta[1] for benchmarking. The UE was OnePlus 5T phone tethered to a MacBook via USB, which the MacBook used as a modem.

The detailed datasets, software versions, and hardware component specifications are available on Github[2]. Given the results, further research is inevitable to debug anomalies such as the first request latency of an empty webpage of 65.6ms as shown in Fig. 4. While WiFi does yield better performance, it is worth noting that LTE has wider operation range thus is more suitable for applications requiring high mobility. Further, we note the EPC uses userland implementations of the GPRS Tunneling Protocol (GTP), and Stream Control Transmission Protocol (SCTP), which are the transport protocols LTE communication is based on. As both protocols have Linux kernel implementations, the source code of the EPC could be modified to use the more privileged versions thus facilitating higher performance and lower latency. Moreover, integrating Data Plane Development Kit bindings would provide further network performance gains [7].

TABLE II: Latency and throughput

Test	Mean	99th	Throughput
empty HTML document	20.6ms	45.9ms	9.38MB/s
1MB blob	33.6s	40.5s	6.72MB/s
10MB blob	332.5s	336.0s	7.52MB/s
10MB blob on WiFi	62.5s	68.7s	38.65MB/s

[1]https://github.com/tsenart/vegeta
[2]https://github.com/toldjuuso/haavisto2019infra

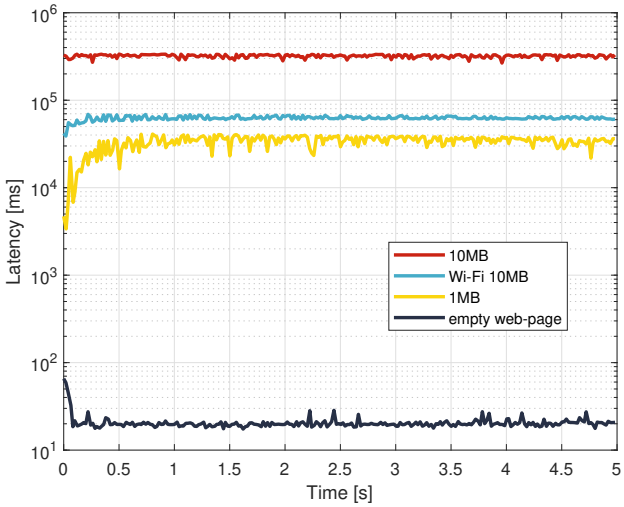

Fig. 4: End-to-end latencies on four different data sets.

V. Discussion

In this paper, we have presented an OSS-based infrastructure for development and deployment of distributed OTA applications for MEC atop 5G networks. In essence, we provide low latency computation resources for end-user applications. In comparison with previous work, we present the first RAN infrastructure combining OTA, DevOps and PaaS. We observed that the network can handle the latency of 21ms and bandwidth of 7.5MB/s in E2E OTA testing. Introducing DevOps methodology can lower the learning curve to create 5G MEC services for end-user application developers. Moreover, the OSS architecture allows continuation research of off-the-spec RAN modifications which may be next to impossible to implement in proprietary environments.

The contribution stitches together interfaces of telecommunications and computer science through software engineering. As a result, we formed a mobile network operator, which correspondingly provides edge computing in a similar manner as industry PaaS businesses offer cloud computing. The authors are not aware of any similar pre-existing counterparts being presented in literature, and hence the research is novel. We continue to conduct practical research in the presented architecture. Thus, this pragmatic instance helps to envision the role of future mobile network operators while retaining a level of lucidness in research through pragmatism.

Regarding virtualization, the current architecture requires hardware to support Docker to act as a schedulable host. Another possibility would be to use Wasm [46] as the application development environment, with reduced communication and computational overhead stipulating virtualization change from Docker to Google V8. V8 supports more processor architectures than Docker, which would enable a broader range of computational units to join the distributed platform.

The presented hardware platform could also be improved, targeting enabling implementations of deadline-critical NFV functionality using open-source hardware such as Power9 architecture and Open Compute Project network hardware. These implementations could reside on more profound levels of software abstraction, thus resulting in better performance and topics of future research.

Finally, the University of Oulu owns the RAN hardware, and has the regulated spectrum licenses to deploy the network OTA. This enables a future scenario with eSIMs, in which the campus area has an open cellular network for students and faculty members. In such scenario, subscribers could programmatically have SIM credentials created using QR codes on the campus hallways while aspiring developers could deploy software both as RAN infrastructure components, but also as end-user E2E applications accessible over-the-air within the campus periphery.

VI. Conclusion

An infrastructure for RAN was presented, where the target is to enable low latency connections for UEs in the context of 5G networks for MEC. Our evaluation shows that the network, at the premises of the University of Oulu, can handle latency of 21ms and bandwidth of 7.5MB/s in E2E OTA testing. These results provide a real-world baseline for further improving such infrastructures towards the goals of 5G and edge computing. The OSS infrastructure allows off-the-spec RAN modifications, this is difficult to realize in proprietary environments. Our future work addresses improvements concerning the EPC and edge host hardware. Moreover, we envision further real-world OTA application deployments and studies with industry developers, researchers, and students on the presented open cellular network.

VII. Acknowledgements

This research is financially supported by Academy of Finland 6Genesis Flagship (grant 318927) and by the AI Enhanced Mobile Edge Computing project, funded by the Future Makers program of Jane and Aatos Erkko Foundation and Technology Industries of Finland Centennial Foundation.

References

[1] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[2] M. Condoluci, G. Araniti, T. Mahmoodi, and M. Dohler, "Enabling the iot machine age with 5g: Machine-type multicast services for innovative real-time applications," *IEEE Access*, vol. 4, pp. 5555–5569, 2016.

[3] M. Mezzavilla, M. Zhang, M. Polese, R. Ford, S. Dutta, S. Rangan, and M. Zorzi, "End-to-end simulation of 5g mmwave networks," *IEEE Communications Surveys & Tutorials*, 2018.

[4] P. Ahokangas, S. Moqaddamerad, M. Matinmikko, A. Abouzeid, I. Atkova, J. F. Gomes, and M. Iivari, "Future micro operators business models in 5g," *The Business & Management Review*, vol. 7, no. 5, p. 143, 2016.

[5] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Migrating to cloud-native architectures using microservices: an experience report," in *European Conference on Service-Oriented and Cloud Computing*, pp. 201–215, Springer, 2015.

[6] M. Amaral, J. Polo, D. Carrera, I. Mohomed, M. Unuvar, and M. Steinder, "Performance evaluation of microservices architectures using containers," in *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*, pp. 27–34, IEEE, 2015.

[7] C.-N. Mao, M.-H. Huang, S. Padhy, S.-T. Wang, W.-C. Chung, Y.-C. Chung, and C.-H. Hsu, "Minimizing latency of real-time container cloud for software radio access networks," in *Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on*, pp. 611–616, IEEE, 2015.

[8] J. Larisch, J. Mickens, and E. Kohler, "Alto: lightweight vms using virtualization-aware managed runtimes," in *Proceedings of the 15th International Conference on Managed Languages & Runtimes*, p. 8, ACM, 2018.

[9] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," *Queue*, vol. 14, no. 1, p. 10, 2016.

[10] M. Richards, *Microservices vs. service-oriented architecture*. O'Reilly Media, 2015.

[11] N. Dmitry and S.-S. Manfred, "On micro-services architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, 2014.

[12] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "Softran: Software defined radio access network," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 25–30, ACM, 2013.

[13] Q. Xu, J. Huang, Z. Wang, F. Qian, A. Gerber, and Z. M. Mao, "Cellular data network infrastructure characterization and implication on mobile content placement," in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pp. 317–328, ACM, 2011.

[14] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An Untold Story of Middleboxes in Cellular Networks," *SIGCOMM Computer Communication Review*, vol. 41, pp. 374–385, Aug. 2011.

[15] T. Killalea, "The hidden dividends of microservices," *Communications of the ACM*, vol. 59, no. 8, pp. 42–45, 2016.

[16] C. Vallati, A. Virdis, E. Mingozzi, and G. Stea, "Exploiting lte d2d communications in m2m fog platforms: Deployment and practical issues," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pp. 585–590, IEEE, 2015.

[17] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: issues and challenges," *Ieee Network*, vol. 30, no. 4, pp. 46–53, 2016.

[18] Y. Elkhatib, B. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir, and E. Rivière, "On using micro-clouds to deliver the fog," *arXiv preprint arXiv:1703.00375*, 2017.

[19] H.-C. Hsieh, C.-S. Lee, and J.-L. Chen, "Mobile edge computing platform with container-based virtualization technology for iot applications," *Wireless Personal Communications*, vol. 102, no. 1, pp. 527–542, 2018.

[20] R. Muñoz, R. Vilalta, N. Yoshikane, R. Casellas, R. Martínez, T. Tsuritani, and I. Morita, "Integration of iot, transport sdn, and edge/cloud computing for dynamic distribution of iot analytics and efficient use of network resources," *Journal of Lightwave Technology*, vol. 36, no. 7, pp. 1420–1428, 2018.

[21] K. Amemiya, Y. Akiyama, K. Kobayashi, Y. Inoue, S. Yamamoto, and A. Nakao, "On-site evaluation of a software cellular based mec system with downlink slicing technology," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, pp. 1–7, IEEE, 2018.

[22] L. T. Bolivar, C. Tselios, D. M. Area, and G. Tsolis, "On the deployment of an open-source, 5g-aware evaluation testbed," in *2018 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pp. 51–58, IEEE, 2018.

[23] J. S. Walia, H. Hämmäinen, and M. Matinmikko, "5g micro-operators for the future campus: A techno-economic study," in *Internet of Things Business Models, Users, and Networks, 2017*, pp. 1–8, IEEE, 2017.

[24] Y. Siriwardhana, P. Porambage, M. Liyanage, J. S. Walia, M. Matinmikko-Blue, and M. Ylianttila, "Micro-Operator driven Local 5G Network Architecture for Industrial Internet," *arXiv preprint arXiv:1811.04299*, 2018.

[25] C.-W. Tseng, Y.-K. Huang, F.-H. Tseng, Y.-T. Yang, C.-C. Liu, and L.-D. Chou, "Micro Operator Design Pattern in 5G SDN/NFV Network," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[26] T. Kämäräinen, M. Siekkinen, A. Ylä-Jääski, W. Zhang, and P. Hui, "A measurement study on achieving imperceptible latency in mobile cloud gaming," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, pp. 88–99, ACM, 2017.

[27] F. Eckermann, P. Gorczak, and C. Wietfeld, "tinyLTE: Lightweight, Ad-Hoc Deployable Cellular Network for Vehicular Communication," *arXiv preprint arXiv:1802.09262*, 2018.

[28] I. Hadžić, Y. Abe, and H. C. Woithe, "Edge computing in the epc: a reality check," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, p. 13, ACM, 2017.

[29] C.-Y. Li, H.-Y. Liu, P.-H. Huang, H.-T. Chien, G.-H. Tu, P.-Y. Hong, and Y.-D. Lin, "Mobile edge computing platform deployment in 4g LTE networks: A middlebox approach," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[30] C. Streiffer, A. Srivastava, V. Orlikowski, Y. Velasco, V. Martin, N. Raval, A. Machanavajjhala, and L. P. Cox, "eprivateeye: to the edge and beyond!," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, p. 18, ACM, 2017.

[31] M. Carmo, S. Jardim, A. Neto, R. Aguiar, D. Corujo, and J. J. Rodrigues, "Slicing wifi wlan-sharing access infrastructures to enhance ultra-dense 5g networking," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.

[32] G. Cattaneo, F. Giust, C. Meani, D. Munaretto, and P. Paglierani, "Deploying cpu-intensive applications on mec in nfv systems: The immersive video use case," *Computers*, vol. 7, no. 4, p. 55, 2018.

[33] E. Piri, P. Ruuska, T. Kanstrén, J. Mäkelä, J. Korva, A. Hekkala, A. Pouttu, O. Liinamaa, M. Latva-Aho, K. Vierimaa, *et al.*, "5gtn: A test network for 5g application development and testing," in *Networks and Communications (EuCNC), 2016 European Conference on*, pp. 313–318, IEEE, 2016.

[34] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva, "Vr is on the edge: How to deliver 360 videos in mobile networks," in *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pp. 30–35, ACM, 2017.

[35] P. Karimi, M. Sherman, F. Bronzino, I. Seskar, D. Raychaudhuri, and A. Gosain, "Evaluating 5g multihoming services in the mobilityfirst future internet architecture," in *Vehicular Technology Conference (VTC Spring), 2017 IEEE 85th*, pp. 1–5, IEEE, 2017.

[36] A. Gosain, M. Berman, M. Brinn, T. Mitchell, C. Li, Y. Wang, H. Jin, J. Hua, and H. Zhang, "Enabling campus edge computing using geni racks and mobile resources," in *Edge Computing (SEC), IEEE/ACM Symposium on*, pp. 41–50, IEEE, 2016.

[37] A. Van Kempen, T. Crivat, B. Trubert, D. Roy, and G. Pierre, "Mec-conpaas: An experimental single-board based mobile edge cloud," in *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2017 5th IEEE International Conference on*, pp. 17–24, IEEE, 2017.

[38] E. McLaughlin, "In cloud software wars, mesosphere bows to kubernetes." https://www.theinformation.com/articles/in-cloud-software-wars-mesosphere-bows-to-kubernetes, September 2017.

[39] ETSI, *ETSI GS MEC*, 3 2016. V1.1.1.

[40] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[41] CoreOS, "CoreOS Container Linux Update Strategies." https://coreos.com/os/docs/2037.0.0/update-strategies.html, Jan. 2019.

[42] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, "NetBricks: Taking the V out of NFV.," in *OSDI*, pp. 203–216, 2016.

[43] A. Jain, N. Sadagopan, S. K. Lohani, and M. Vutukuru, "A comparison of sdn and nfv for re-designing the lte packet core," in *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*, pp. 74–80, IEEE, 2016.

[44] L. Gauthier, "4G Core Network Training." https://www.openairinterface.org/docs/workshop/5_OAI_Workshop_20180620/Training/GAUTHIER_Core_Network_Training_Gauthier_Eurecom.pdf, June 2018.

[45] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: an open-source platform for LTE evolution and experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, pp. 25–32, ACM, 2016.

[46] A. Haas, A. Rossberg, D. L. Schuff, B. L. Titzer, M. Holman, D. Gohman, L. Wagner, A. Zakai, and J. Bastien, "Bringing the web up to speed with webassembly," in *ACM SIGPLAN Notices*, vol. 52, pp. 185–200, ACM, 2017.