Performance Evaluation of Secured Containerization for Edge Computing in 5G Communication Network

MOHAMMED, Abubakar Saddiq [1], MOSUDI, Isiaka Olukayode [1] and ZUBAIR, Suleiman [1]

[1] Telecommunication Engineering Department, Federal University of Technology, Minna. Niger State, Nigeria. abu.sadiq@futminna.edu.ng

Abstract. Portable mobile devices capabilities are bedeviled by shortfall in computational memory, storage, networking and energy carrying capacity. These issues of resource poverty have been challenges the telecommunication industry had to cope with in portable mobile devices. Earlier research efforts like Cloudlets, Cyber Foraging, Mobile Cloud Computing (MCC), and recently, Multi-access Edge Computing (MEC) have been proffered. Container workflow orchestration provides automation capabilities for MEC to deliver just in time applications and services. This enables service providers to give ubiquitous access to specified applications or services since applications infrastructures are pushed to repositories and can be deployed by downloading from a repository. Containers are more easily orchestrated unlike virtual machines (VM) or bare metal, this provides the required DevOps ecosystem for developers and engineers to work across the entire application lifecycle. This streamlines the process of automating application setup by establishing standard procedures for rapid deployments while also reducing human error associated with manual setups. In this paper, secured containerization techniques were employed in order to achieve end-to-end low latency figures for MEC infrastructure isolation as required by the European Technology Standard Institute (ETSI) for MEC application security. An evaluation was done by comparing secured containerized app deployment in the cloud and edge scenarios. The result obtained validated that edge computing has lower User Plane (UP) latency figures, less backhaul traffic and a lower application failure rate.

Keywords: Kata Container, Multi-access Edge Computing, User Plane Latency.

1 Introduction

The deployment of 5G communication standards is a precursor to the explosive evolution of Information and Communication Technology (ICT) innovations for mobile devices. Aggregation and integration of wide-range of applications and operations such as: Machine-to-Machine (M2M) Communication, Internet of Things (IoT), emerging Vehicle

Technologies, Virtual Reality (VR), Augmented Reality (AR), etc are at its wake. With these comes a correspondingly large increase in the number of smart mobile devices to approximately over 50 billion. However, this number is small compared to the exponential growth in the volume of data generated by these powerful applications and feature-rich contents. This will create a hype for mobile data traffic and high computing requirements according to Skarpness [1]. As a result, constraints of computational resources and network resources are envisaged for cellular mobile communication User Equipment (UE).

To resolve these problems, the computational requirements of mobile applications can be offloaded to tethered external infrastructures with adequate resources for processing. Different interventions have been proposed, which include Cyber Foraging, Cloudlet, Mobile Cloud Computing (MCC) and Multi-access Edge Computing (MEC), Mosudi et al. [2].

2 Containerization

Containerization is the process by which the Operating System (OS) kernel allows running of isolated user-space instances called containers. These are standard collection of software that bundles up the code and all its dependencies together as an abstraction at the OS application layer, thereby enabling the application to run quickly and reliably from one computing environment to another. Container images are typically tens of MBs in size. It can handle more applications and require fewer machines and OS, Adufu et al. [5]. Containers compared to Virtual Machines (VM) are more suitable for MEC for the sake of storage limitation and computing resources optimization. Containerization allows hardware resources to be decoupled from software, enabling packaged software to execute on multiple hardware architectures providing several benefits such as rapid construction, instantiation, and initialization of virtualized instances, Taleb et al. [3].

MEC resources can be allocated to containers for better isolation, performance and allowing for easy collaboration and deployment of applications across different mobile environments, Willis [6]. Orchestrated containerized MEC will provide efficient infrastructures needed for migration of monolith legacy applications onto 5G service platforms. This enables the breaking down of large applications into micro-service deployable on a large number of interconnected MEC platforms, Alam et al. [4]. The containerization ecosystem has become so matured, presenting a whole lot of its orchestrators such as Docker Swan, Kubernetes (k8s), Marathon, Amazon container engine. Google Container Engine (GKE), and Azure container service, Kata [7], Piparo et al.[8], Sanchez [9], Hoque et al. [10] and Augustyn and Warchal [11]. The test bed experiments in the paper made use of Kata-containers.

2.1 Kata Container

Kata containers run in dedicated kernels to provide isolation of network input/output, memory and can utilize hardware-enforced isolation with virtualization extensions. However, it is backward compatible with industry standards such as Open Containers Initiatives (OCI) container format, Kubernetes Container Runtime Interface (CRI), as well as legacy virtualization technologies while consistent with standard Linux containers in performance. Kata is based on the Kernel Virtual Machine (KVM) hypervisor with an option for Quick Emulator/Net Emulator (QEMU/NEMU). NEMU is actually a stripped-down version of QEMU by removing emulation not required thereby reducing the attack surface. It is more secure than a traditional container by replacing default container runtime (runC) with Kata-runtime. Relying on Kata-agent, shim for I/O while running Kata runtime instead of runC container runtime as available in Docker. Kata containers are light and fast containers.

3 Design of MEC Deployment Scenario

In this section, the 5G network 3GPP and non-3GPP transport components specifications were evaluated, and models for MEC deployment scenarios for 5G network were designed. This was carried out to provide the platform to compare MEC application end-to-end transport latency in 5G and 4G deployments. This work leveraged on control/user plane separation (CUPS), Lower Layer Functional Splits (LLFS), and Higher Layer Functional Splits (HLFS) and 3GPP 5G Service-Based Architecture (SBA), ESTI [12], and distributed Common Compute Platform (CCP) which permits the location of Virtualized Network Functions (VNFs) in different parts of the network for management of different capabilities. MEC hosts were located at the Centralized Unit (CU) connected directly to the Packet Data Convergence Protocol (PDCP) thereby, reducing the estimated total UP latency for MEC deployment in 5G.

The end-to-end transport latency has an effect on determining the value of UP latency, and in combination with CP latency, determines the effective QoE. Higher Layer Functional Split (HLFS) option 2 for the mid-haul and Lower Layer Functional Split (LLFS) option 7 for front-haul will permit four RAN deployment scenarios and thus four MEC deployment scenarios (Fig. 1),

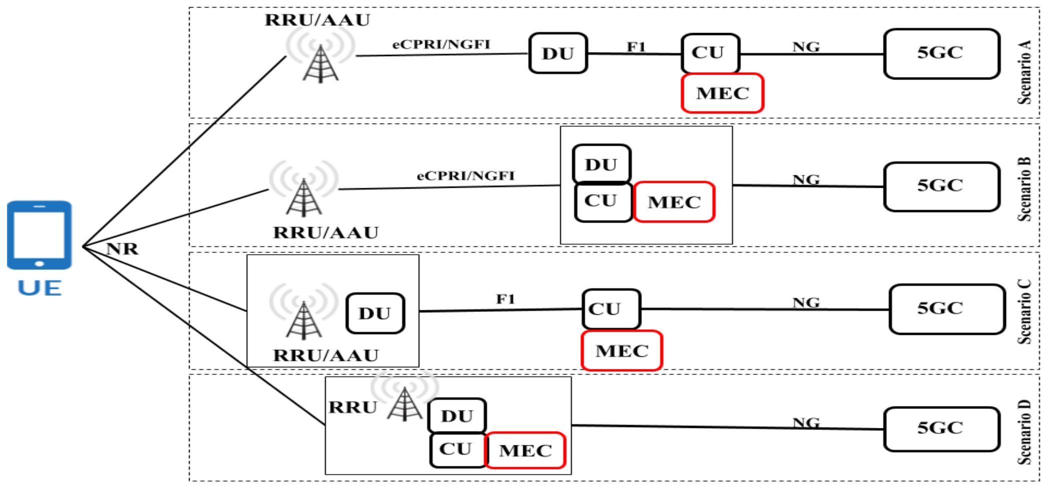

Fig. 1. 5G MEC deployment models (SBA RAN)

3.1 Latency Evaluation

The latency figure, T can be obtained using the relationship for the total one-way UP latency for an application deployed on 4G/LTE,

$$T = T_{Radio} + T_{Backhaul} + T_{Core} + T_{Transport} \qquad (1)$$

by modifying (1) for LTE/EPC, we obtain

$$T = T_{LTE} + T_{EPC} + T_{Transport} \qquad (2)$$

where,
$\mathbf{T_{LTE}}$ = one-way packet propagation delay between UE and eNB, plus packet processing time; $\mathbf{T_{EPC}}$ = one-way packet propagation delay between eNB and EPC, plus processing delay with the core network; $\mathbf{T_{Transport}}$ = one-way packet propagation between the EPC and Packet Data Network (PDN). This might include propagation delay to the internet, if service requested by the UE has to be sourced from the Internet.

Latency values will vary from one MEC deployment scenario to another. Quantifying all the parameters is challenging due to differences in the performance of equipment. Considering 5G end-to-end network from DU to CU, and MEC host. However, we assumed a 1-way latency range between 5ms and 8ms between CU and DU and in essence, 8ms network latency between CU and DU eases the co-location of the CU with MEC.

Therefore, the total one-way user plane latency becomes,

$$T = T_{NR} + T_{DU} + T_{CU} + T_{Transport} \qquad (3)$$

where,
$\mathbf{T_{NR}}$ = the one-way packet propagation for New Radio (NR) delay between UE and DU, plus packet processing time; $\mathbf{T_{DU}}$ = one-way packet propagation delay between DU and CU, plus processing delay with the CU; $\mathbf{T_{CU}}$ = one-way packet propagation delay between CU and 5GC, plus processing delay within the 5GC; $\mathbf{T_{Transport}}$ = one-way packet propagation between the 5GC and DN. This might include propagation delay to the Internet if the service requested has to be sourced from the Internet.

Deployment of MEC in all the four scenarios in the model above provided the option for a direct connection between MEC and the CU. The total one-way user plane latency becomes,

$$T = T_{NR} + T_{DU} + T_{CU} \qquad (4)$$

There is a need for a 5G-capable integrated development environment in the quest to investigate the deployment of MEC at the 5G CU, but we could not get a simulator for this purpose. Instead, leaning on Docker containers, Kata-runtime, and OS-builder - Kata containers guest OS building scripts, a sandbox application was built to gain insight into the advantages of computing at the network edge compared to at the remote cloud servers. In order to compare MEC versus MCC deployments of resource-intensive applications, a mobile web application was built, shipped in a secure container image, saved as a code and pushed to a repository. This combination of definition files were deployed on Docker engines hosted in the remote cloud and edge servers. The chosen target was the web application platform because of its capabilities of execution on a wide range of devices and mobile environments without modification of the application codebase.

3.2 Experimental Environment

Experimental environment incorporated Ubuntu server, Ubuntu Docker image and Docker container engine with its default runtime (runC), replaced with Kata-Runtime. The Ubuntu Kata-container image was created using OS-builder. Python programming language was used for application logic, dataset result generation, cleaning and graphing. The test application backend was based on Python Flask micro web framework while the front-end was built using HTML/CSS/JavaScript. Docker was used for application shipping and Locust framework for load testing. Publicly available Atlassian Bitbucket git and Docker Hub repositories were used for web application code base and container image repositories respectively. The containerized mobile application was deployed using Docker, but Kata-runtime replaced runC to ensure app isolation at the kernel level. This ensured the deployment of the MEC application at the speed of containers while maintaining the security available in VMs. The test mobile application was a memory and processor-intensive mobile web application that generates Rubik cubes images and provides a breakdown of the cube details. These details were total cubelets faces, cubelets and hidden cubelets. It also holds the generated graphics in the memory while rendering it on the end-user devices. This is comparable to graphics generation and rendering in mobile game applications.

4 Experimental Results and Discussion

There were two sets of results obtained. The first set was obtained from the application of 5G transport interface specifications on the four proposed MEC deployment models. Evaluating equation (4) for eMBB proposed four deployment scenarios shown in Fig.1. by applying 5G specification values Mosudi et al. [2];

4.1 Deployment Scenario

Scenario A : Optimally this prototype produced an estimated round-trip time (RTT) value of 11.2ms. Using Equation (4),

$$T = T_{NR} + T_{DU} + T_{CU}$$

$$\begin{aligned} \text{Minimum } T &= 4000 + 100 + 1500\ \mu\text{sec} \\ &= 5.6 \times 10^{-3}\ \text{s} \\ \text{Maximum } T &= 4000 + 100 + 10000\ \mu\text{sec} \\ &= 14.1 \times 10^{-3}\ \text{s} \end{aligned}$$

Scenario B : Optimally, the RTT is 8.2ms.

$$\begin{aligned} T &= 4000 + 100\ \mu\text{sec} \\ &= 4.1 \times 10^{-3}\ \text{s} \end{aligned}$$

Scenario C : Optimal RTT is 11ms.

$$\begin{aligned} \text{Minimum } T &= 4000 + 1500\ \mu\text{sec} \\ &= 5.5 \times 10^{-3}\ \text{s} \\ \text{Maximum } T &= 4000 + 10000\ \mu\text{sec} \\ &= 14 \times 10^{-3}\ \text{s} \end{aligned}$$

Scenario D : Optimal RRT is 8ms.

$$\begin{aligned} T &= 4000\ \mu\text{sec} \\ &= 4 \times 10^{-3}\ \text{s} \end{aligned}$$

4.2 Load Test

The second set of results were obtained from the experimental load test of the secured containerized mobile web application deployed on remote cloud location and servers. Fig. 2. depicts the MCC Test Scenario.

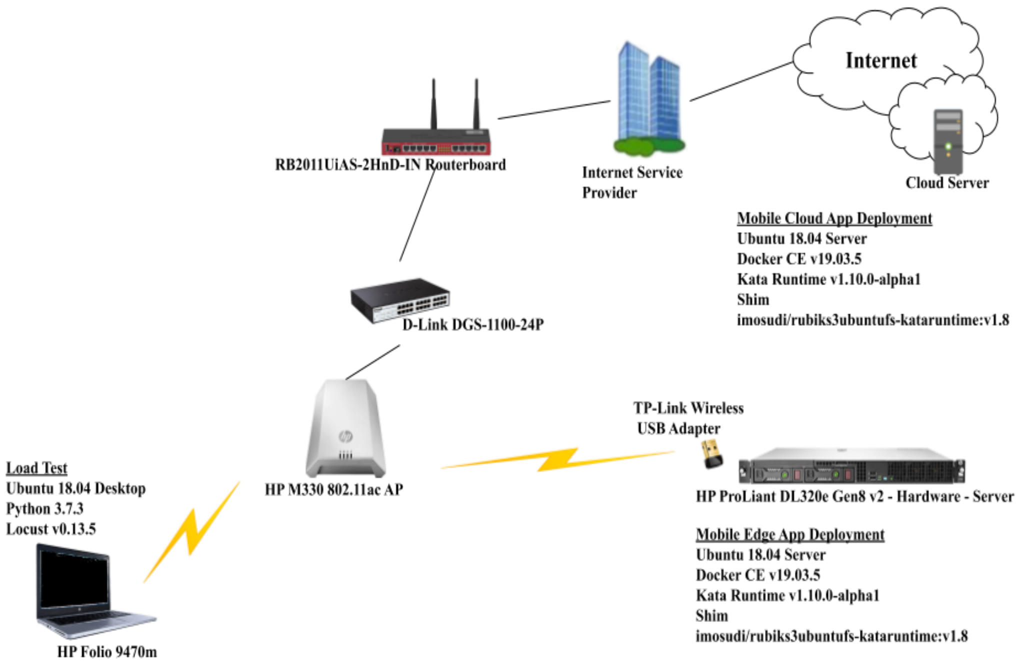
Internet
RB2011UiAS-2HnD-IN Routerboard
Internet Service Provider
Cloud Server
Mobile Cloud App Deployment
Ubuntu 18.04 Server
Docker CE v19.03.5
Kata Runtime v1.10.0-alpha1
Shim
imosudi/rubiks3ubuntufs-kataruntime:v1.8
D-Link DGS-1100-24P
TP-Link Wireless USB Adapter
Load Test
Ubuntu 18.04 Desktop
Python 3.7.3
Locust v0.13.5
HP M330 802.11ac AP
HP ProLiant DL320e Gen8 v2 - Hardware - Server
Mobile Edge App Deployment
Ubuntu 18.04 Server
Docker CE v19.03.5
Kata Runtime v1.10.0-alpha1
Shim
imosudi/rubiks3ubuntufs-kataruntime:v1.8
HP Folio 9470m

Figure 2. MCC Test Scenario

The results obtained from the experiments included; the total requests per second (req/s) made to the application deployed, request time stamps, requests failure per second (req/s), minimum, median and maximum application response time, average application data download size, 50 percentile and 95 percentile application response time among other result parameters. All these were determined for both edge site and cloud application deployments respectively. Each time, the experiment lasted about 2 hours. It was observed that the application response time and the amount of downloaded application data follow the same pattern corresponding to the size of the Rubik's cube being rendered. Likewise, failures were more prevalent with the MCC deployments compared to the relatively stable MEC deployments.

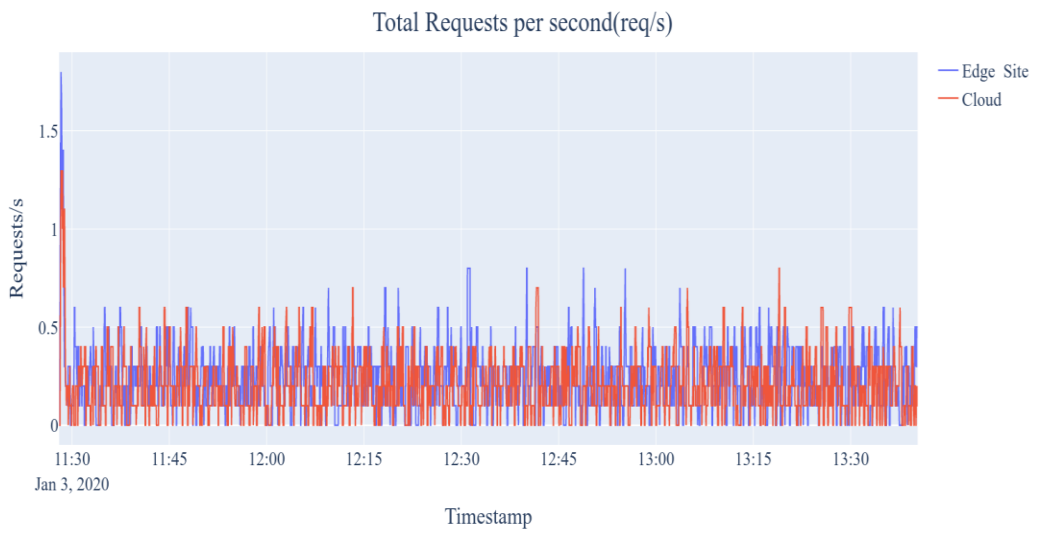

Fig. 3. Total Request per sec

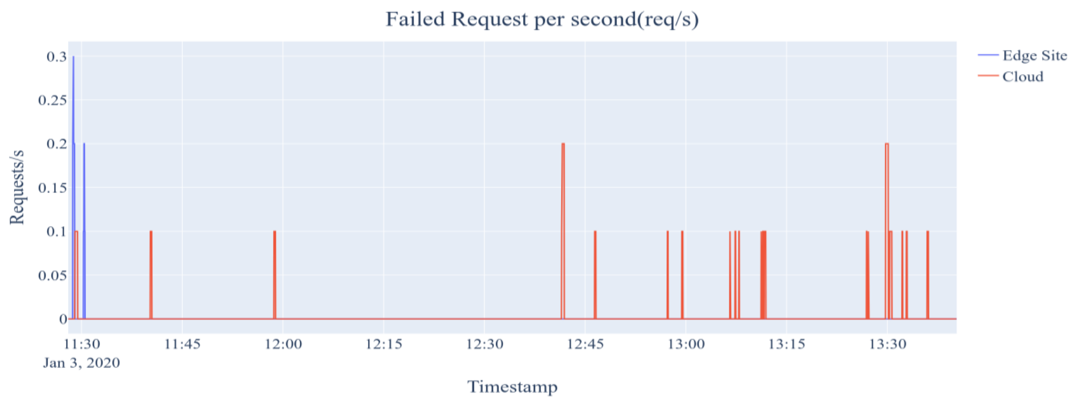

Fig. 4. Total Failed Request rate

Request rate for both the edge site and cloud deployments are presented in the results. The intention to simulate a random Rubik's cube size between values of three and fourteen did pay off. The requested rate for both deployments were about the same as indi cated in Fig 3., which justifies a fair comparison. Fig. 4 shows that failures were more prevalent in cloud deployments. Application maximum and median response time for both edge and cloud deployments were presented in Fig. 5.

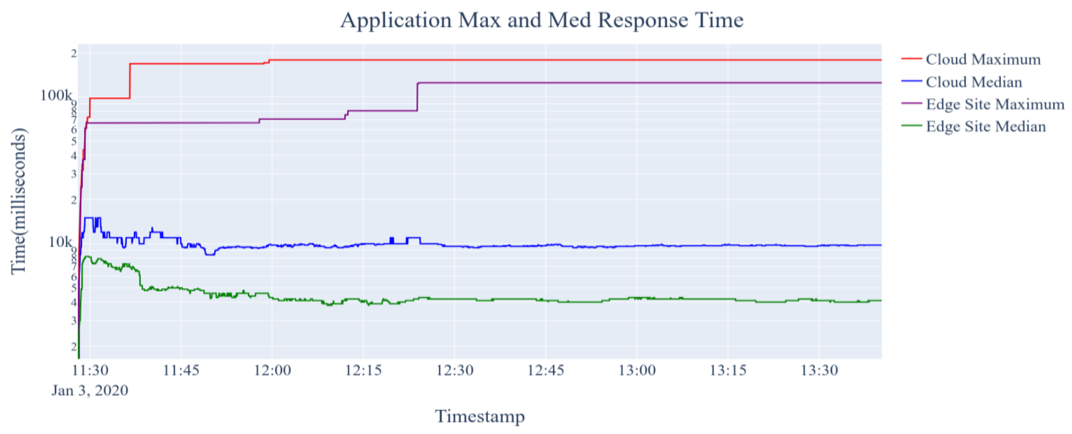

Fig. 5. Application Max. and Medium Response Time

The average application data download are presented in Fig. 6. The same application version was deployed for both scenarios and with the convergence in the application request rates as shown in Fig. 3. These had a significant effect on the amount of application data downloaded for both scenarios during the experiments. The application average data download also converged, validating fair comparison. The application response for half of the experiment duration, 50 percentile, second quartile or median response time is plotted in Fig 7.

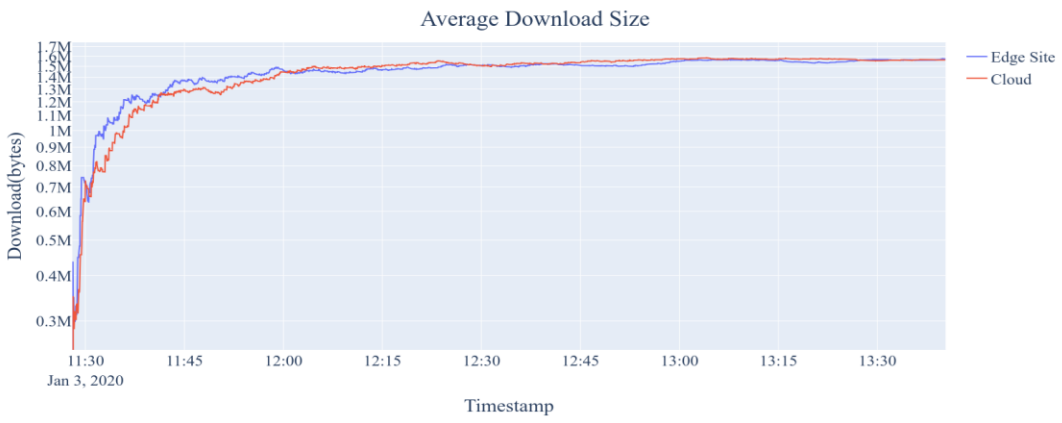

Fig. 6. Average Application Content Download in Bytes

The general application response time over a period of two hours for 50 percentiles is discussed below. The application response time for less tha 95 per cent of the time span of the experiment and 95 per cent is presented in Fig 7 and Fig 8.

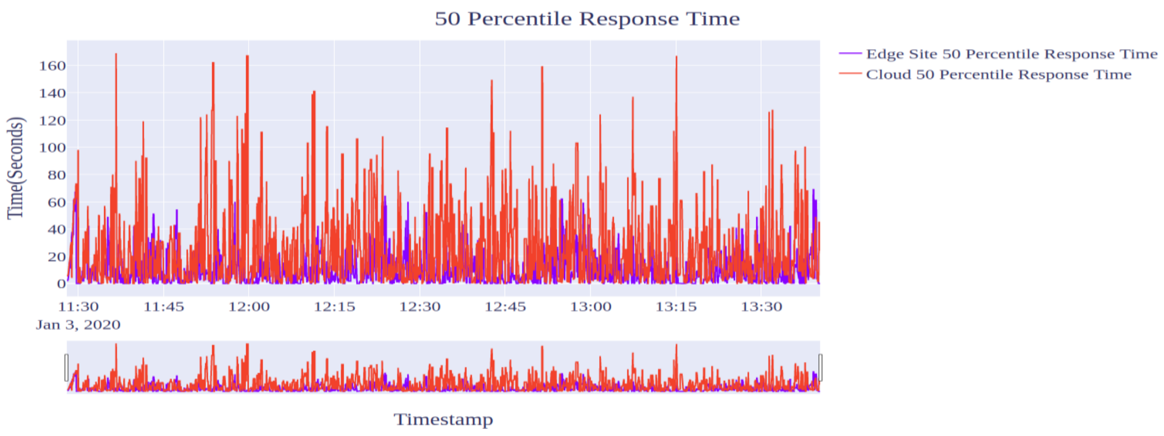

Fig. 7. Second Quartile Application Response Time

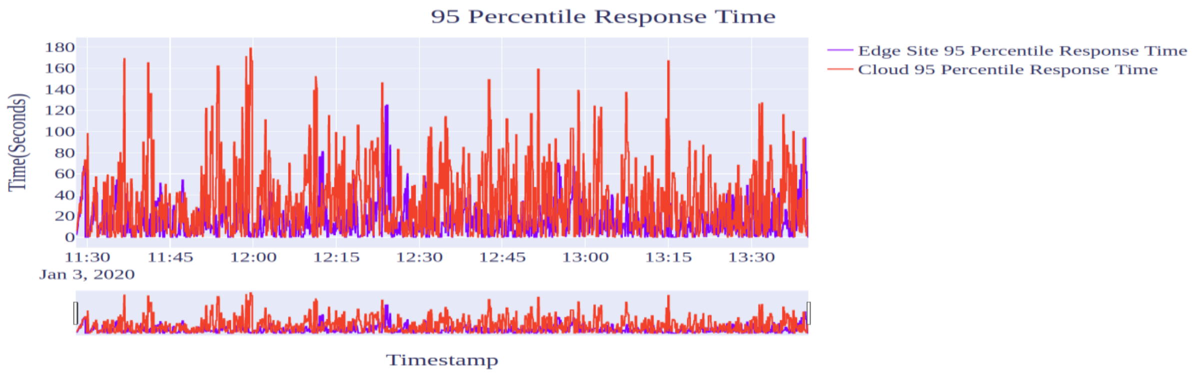

Fig. 8. 95 Percentile of Response Time

The round-trip times (RTT) from the MEC deployment are all within the latency requirements for VR and AR of 7-12ms. Tactile Internet is < 10ms, Vehicle-to-Vehicle is < 10ms, Manu facturing and Robotic, Control/Safety Systems are between 1ms and 10ms. The attendant savings in compute resources are part of the reasons for the choice of containers over VM.

Considering a production deployment scenario with several thousand geographically dispersed UEs connected to a mobile application hosted at a remote cloud data center or within the operator DN, this will create high bandwidth traffic and exert serious penalty on the operator backhaul network. However, in the proposed edge site deployment scenario, UEs will take advantage of consuming MEC applications hosted at the network edge, deployed at the CU thereby removing the issue of heavy data traffic which can result in bottleneck on the backhaul networks.

Secondly, both median and maximum application response time were considered for both edge sites and cloud deployments as indicated in Fig. 5. It was observed that in the maximum latency figures for the cloud deployment required for smooth running of mobile applications, there were initial failures reported for both deployment at application startup. In fact, the figure for the edge deployment was very poor, but failure finally disappeared. However, there is evidence that there is a higher application failure response rate for cloud deployments compared with the edge site deployment as indicated in Fig. 4. This might have adverse effects on the adoption of new and emerging latency-sensitive applications. The maximum edge response time for edge deployment was initially a little above 60sec compared to about 170 sec for cloud deployment as shown in Fig. 6. This test was not a 5G network but it was obvious that deployment on a real 5G network with adequate MEC server resources can normalize the edge figures to more acceptable values. Furthermore, the comparison of minimum application response time confirms the proposal for containerized applications deployment at the edge for 5G networks.

Clearly, edge deployment response time is more visible looking around low latency values, unlike the cloud which dominates the graph skyline indicating consistent unacceptably high latency values. The fact that containers are secure and deployable for MEC infrastructures will increase the ability of enterprise developers by improving collaboration to quickly deliver scalable and reliable applications and services at paces required for 5G rollout not jeopardizing the security of the end-to-end network. Containers can provide the required DevOps ecosystem for developers and engineers to work across the entire application lifecycle, from designs, development, testing to deployment and operations.

5 Conclusion

Secured containerization technique was used to achieve end-to-end low latency figures with MEC infrastructure isolation and application security. The secured containerized application was deployment in both the cloud and edge scenarios, it was validated that the edge scenario has lower User Plane latency figures, less backhaul traffic and a lower application failure rate. Secured containers using Docker and Kata containers provide most of the essential features required for MEC infrastructures making it suitable for resource-intensive mobile applications speed and isolation requirements to guarantee safety within the mobile ecosystem expected with massive deployment of 5G UEs. Container workflow orchestration provides automation capabilities for MEC to deliver just in time applications and services, enabling service providers to provide ubiquitous access to specified applications or services since applications infrastructures are pushed to repositories and can be deployed by downloading from the repository.

6 References

1. Skarpness, M. “Beyond the Cloud: Edge Computing,” Keynote speech at Embedded Linux Conference Europe, Prague, Czech Republic, 2017.
2. Mosudi, I.O., Abolarinwa, J. & Zubair, S. “Multi-Access Edge Computing Deployments for 5G

Networks" In Proceeding, 3rd International Engineering Conference, pp. 472-479, 2019.

3. Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S. & Sabella, D. "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration", IEEE Communications Surveys and Tutorials, vol. 19 no. 3, 1657–1681, 2017.
4. Alam, M., Rufino, J., Ferreira, J., Ahmed, S. H., Shah, N. & Chen, Y. "Orchestration of Micro-services for IoT Using Docker and Edge Computing", IEEE Communications Magazine, vol.56 no. 9, pp. 118–123, 2018.
5. Adufu, T,. Choi, J. & Kim, Y. "Is container-based technology a winner for high performance scientific applications", In proc. of 7th Asia-Pacific Network Operations and Management Sym-posium (APNOMS), pp. 507- 510, 2016.
6. Willis, J. Docker and the Three Ways of DevOps. Docker Blog. [Online] May 26, 2015. https://blog.docker.com/2015/05/docker-three-ways.
7. Kata Containers. (n.d.). Learn: An overview of the Kata Containers project. Retrieved May 17, 2019, from https://katacontainers.io/learn/.
8. Piparo, D., Tejedor, E., Mato, P., Mascetti, L., Moscicki, J. & Lamanna, M. "SWAN: A service for interactive analysis in the cloud" Future Generation Computer Systems, vol. 78, pp. 1071–1078, 2018.
9. Sanchez C. Scaling docker with Kubernetes. Website. Available online at http://www.infoq.com /articles/scaling-docker-with-kubernetes. 35, 2015.
10. Hoque, S., Brito, M. S. D., Willner, A., Keil, O. & Magedanz, T. "Towards Container Orchestration in Fog Computing Infrastructures", 41st Annual Computer Software and Applications Conference (COMPSAC), pp. 294– 299, 2017.
11. Augustyn, D. R. & Warchał, L. "Cloud Service Solving N-Body Problem Based on Windows Azure Platform", In International Conference on Computer Networks, Berlin, Heidelberg, pp. 84-95, 2010.
12. European Telecommunications Standards Institute (ETSI). (2018, July). TS 129 500 - V15.0.0-5G; 5G System; Technical Realization of Service Based Architecture. - ETSI. Retrieved May, 2019,https://www.etsi.org/deiver/etsi_ts/129500_129599/129500/15.00.00_60/ts_129500v50000p.pdf.