Progress in Human Computer Interaction

2021 Volume 1, Issue 1

ISSN: 2630-4635

Validation of Cloud Robotics System in 5G MEC for Remote Execution of Robot Engines

Sewan Gu, Sungkyu Kang, Wonhong Jeong, Hyungil Moon, Hyunseok Yang, Youngjae Kim*

LG Electronics, Seoul, Republic of Korea

Corresponding author: Youngjae Kim, yjae.kim@lge.com

Abstract

We implemented a real-time cloud robotics application by offloading the robot navigation engine over to 5G mobile edge computing (MEC) sever. We also ran a fleet management system (FMS) in the server and controlled the movements of multiple robots at the same time. The mobile robots under the test were connected to the server through a 5G standalone (SA) network. The public 5G network, which is already commercialized, has been temporarily modified to support this validation by the network operator. Robot engines are containerized based on micro-service architecture and have been deployed using Kubernetes – a container orchestration tool. We successfully demonstrated that mobile robots can avoid obstacles in real-time when the engines are remotely running in a 5G MEC server. Test results are compared with the 5G public cloud and 4G (Long-Term Evolution [LTE]) public cloud as well.

Keywords

5G
Mobile edge computing (MEC)
Robot operating system 2 (ROS2)
Navigation
Cloud
Docker
Kubernetes

1. Introduction

Various system configurations, utilizations, and new attempts to utilize cloud robotics (CR) have been discussed in the literature [1,2]. There can be many applications or ways for robots to utilize cloud servers over a network. These include building/distributing robot execution images through the cloud, controlling robots, collecting/loading data, utilizing AI, and real-time robot control. Among them, real-time robot control is a very challenging task even under the infrastructure of gigabit Wi-Fi and 5G communication networks. However, if private 5G is activated in the future and settled as a network with more reliability and stability, it can be fully utilized in smart factories, smart cities, autonomous vehicles, smart ports, etc.

In this paper, we demonstrate that real-time robot control is possible through cloud offloading of robot engines through experiments on 5G commercial networks and mobile edge computing (MEC). We adopted Kubernetes as the platform for the cloud robotics configuration to operate on MEC [2]. As the most popular Docker orchestration platform in recent years, Kubernetes has powerful features such as scalability, load-balancing, and non-stop rolling-update of computer nodes and processes to cope with the increase in traffic due to the increase in users, and it is also possible to control the behavior to meet each purpose through the customization function according to the requirements of each service in a specific domain [3].

By deploying and operating the autonomous navigation engine, one of the robot engines, on the 5G MEC server, we have verified that it is possible to implement a service with the same performance as that of the on-board robot, and we believe that it is possible to significantly reduce the usage of major computational resources such as central processing unit (CPU) and random-access memory (RAM) because it is not necessary to drive the navigation engine onboard the robot.

This paper is organized into four sections. Section 2 describes the overall system configuration, Section 3 describes the experiments and results, and Section 4 describes future research directions.

2. System configuration

The system used in the experiment can be broadly divided into three parts: the robot, the 5G network, and the MEC. The operating system and platform running on the robot are Yocto [4] and Robot Operating System 2 (ROS2), and all robot engines configured in MEC were dockerized for preliminary verification. The 5G modem was built in-house by LG Electronics and utilized a proven model already in use in several applications. In the following Sections 2.1 and 2.2, we will describe the communication network configuration and the implementation method for cloud services.

2.1. Network configuration

In this paper, we conducted experiments in U+'s Private 5G MEC, and the backbone is not an experimental network but a commercial network. The network is configured as shown in (**Figure 1**).

The cloud server side is operated by adopting the One-Box MEC architecture as shown in **Figure 2**, which includes clusters equipped with core network functions and clusters serving application programs. The clusters are Kubernetes clusters, with the master node controlling

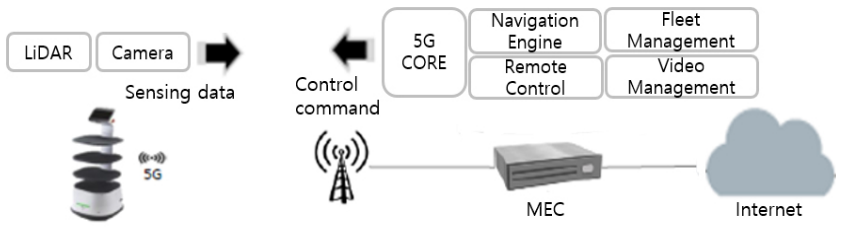


Figure 1. Network architecture

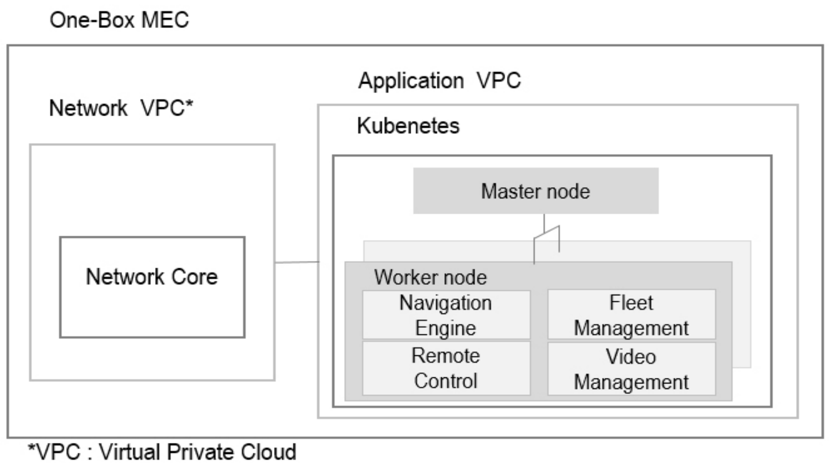


Figure 2. One-box MEC architecture

the entire cluster and worker nodes responsible for launching and operating actual application services. The implementation of the robot engine in the MEC will be discussed in the next section.

While we used a commercial network for the experiment, it is important to note that MEC was activated for a limited period specifically for the experiment. This has also been mentioned in a previous experiment [5].

2.2. Cloud service implementation

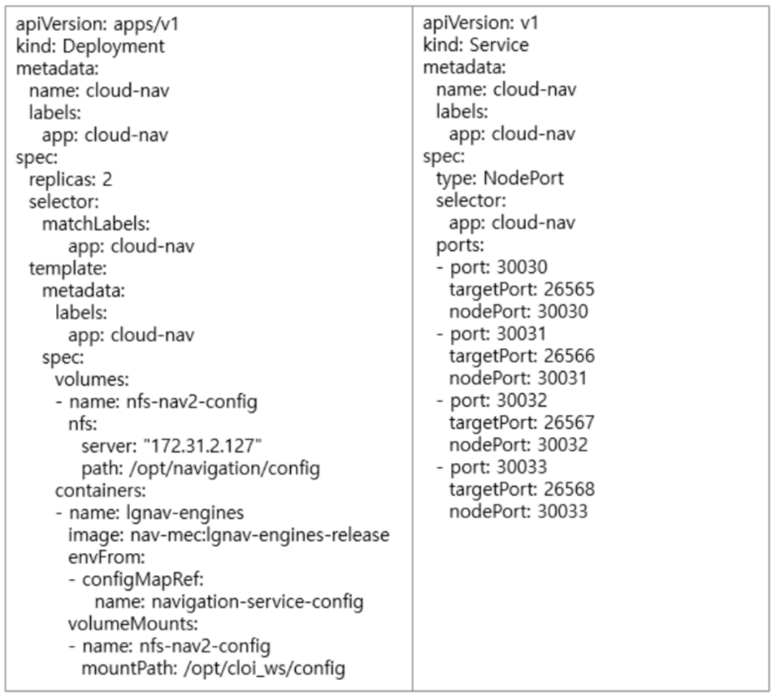
For cloud services, we utilized open-standard virtualization technologies, and as mentioned earlier, we used docker/container technology, which is currently the most widely used technology. It is suitable for microservice architectures, enabling easy distribution and operation of complex structures.

Any service or application program can be packaged as a Docker image, and when such an image is created, it can be executed on the host computer where the Docker engine runs. This technology is very convenient for distributed environments and management. Furthermore, it can operate independently of hardware and operating systems, allowing for rapid deployment of services.

Therefore, in this research, all the individual technologies executing on robots were modularized to the smallest unit and created as Docker images. This allows for independent execution regardless of physical location and enables an architecture where they can interact with each other.

In this study, we ran four robot-related engines on the MEC Kubernetes. They consist of the driving engine, which is the main function; fleet management, for planning multi-robot routes in the same place; remote control, for controlling the robot remotely, monitoring its location, and checking its status; and video management, which outputs video images acquired from the robot's front camera.

In order for these engines to communicate with external robots, Kubernetes network settings had to allow external networking. To achieve this, we configured NodePort as the port for external network communication. NodePort is a Kubernetes service that allows external access to a specific application service through the same port, regardless of which cluster node within Kubernetes it is running on. The deployment/installation method for the driving engine and related information can be seen in **Figure 3** as an example.



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cloud-nav
  labels:
    app: cloud-nav
spec:
  replicas: 2
  selector:
    matchLabels:
        app: cloud-nav
  template:
    metadata:
      labels:
        app: cloud-nav
    spec:
      volumes:
      - name: nfs-nav2-config
        nfs:
          server: "172.31.2.127"
          path: /opt/navigation/config
      containers:
      - name: lgnav-engines
        image: nav-mec:lgnav-engines-release
        envFrom:
        - configMapRef:
            name: navigation-service-config
        volumeMounts:
        - name: nfs-nav2-config
          mountPath: /opt/cloi_ws/config
```

(a)Engine deploy file

```
apiVersion: v1
kind: Service
metadata:
  name: cloud-nav
  labels:
    app: cloud-nav
spec:
  type: NodePort
  selector:
    app: cloud-nav
  ports:
  - port: 30030
    targetPort: 26565
    nodePort: 30030
  - port: 30031
    targetPort: 26566
    nodePort: 30031
  - port: 30032
    targetPort: 26567
    nodePort: 30032
  - port: 30033
    targetPort: 26568
    nodePort: 30033
```

(b)Service deploy file

Figure 3. Desploy .yaml file for Kubernetes

3. Experiment and results

The robots for the experiments are autonomous delivery robots that can be utilized in restaurants. The hardware for motor control and hardware for key application operations on the robots were based on NVIDIA's TX2. The platform used a lightweight Yocto-based Linux operating system, as mentioned in Section 2, and the ROS2 framework [4]. ROS2 is a platform that facilitates easy transmission and reception of various sensors, application messages, and more among robot applications. It also provides various development tools for robot developers and continues to evolve, making it highly regarded in the robotics community.

The most crucial application in this paper's experiment was the autonomous navigation engine. It received light detection and ranging (LiDAR) and camera sensor data from the robot via the 5G network and determined the navigation path based on a predefined map. Therefore, we modularized and dockerized the navigation engine to enable cloud services, as described in Section 2.

In addition, there was a challenge due to the robot and 5G MEC server being in different subnetworks, which prevented ROS2 messages from being exchanged. To overcome this issue, we utilized a developed module called the 'cloud bridge' [6], which allows ROS2 messages between the robot and the server to be transmitted and received as if they were in

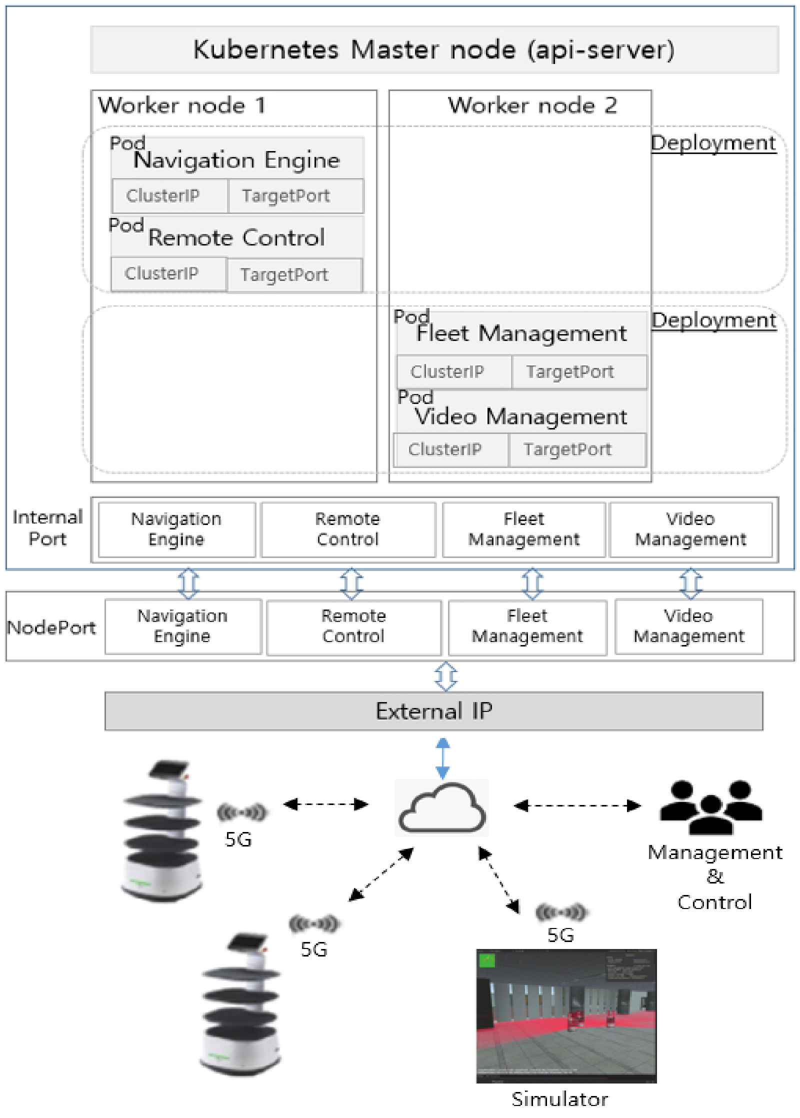


Figure 4. Cloud robotics system on 5G MEC Kubernetes

the same subnetwork. This communication platform is suitable for ROS2 application communication with various external hosts in the 5G network.

The configuration of the cloud robotics system running in the MEC Kubernetes cluster of the service provider is illustrated in **Figure 4**. The experiments were conducted for a certain period at a testbed prepared in Yangjae-dong using U+ 5G trial networks.

3.1. Experimental method

Before conducting experiments on the 5G operational network, we initially conducted verification in a testbed provided by the communication service provider with 5G MEC capabilities. During this phase, instead of using actual robots, we simulated the physical sensors and movements of the robots. We used a 3D simulator that modeled the environment [7]. LiDAR sensor data and motor drive data were sent to the server, and the server sent navigation commands to the simulator to make the robot move. After verifying that basic navigation worked without issues, we proceeded to the actual operation network.

The testbed was assumed to be a restaurant scenario where N autonomous delivery robots provided services to M customer tables. Scheduling, apart from controlling the robots through the navigation engine, was done through cluster control. Real-time video from the cameras mounted on the delivery robots could be monitored through video management. The remote control module allowed confirmation of robot locations and detection of any anomalies or manual commands. Each robot had one dedicated navigation engine, and other engines were independent of the number of robots.

3.2. Experimental results

The packet transmission segments from the robot (terminal) to the 5G core network include robot (terminal) → radio access network (RAN) → user plane function (UPF) → robot engine (microservice) on MEC. In this segment, we first analyzed the average latency of basic 5G and 4G (Long-Term Evolution [LTE]) using the ping test of Internet Control Message Protocol (ICMP), as shown in **Table 1**. This data is related to 56-byte packets used in simple ping tests, without considering the bandwidth required for transmitting a large amount of data like videos.

(1) As expected, we confirmed that the 5G core network is nearly 2.8 times faster than 4G (LTE), and 1.6 times faster in MEC than the 5G public cloud.

(2) When the three-destination driving of two robots for about an hour was performed through scheduling, autonomous driving was performed with the same performance as on-board without any problems, and when comparing the usage of the CPU and RAM in the robot system, we found that it was about 50% more efficient than the existing on-board system, as shown in **Table 2**. The sensor data of the robot utilized for autonomous driving were camera and LiDAR, and the camera

Table 1. Round trip time measured in robot

	5G MEC	**5G public-cloud**	**4G (LTE) public-cloud**
RTT (ms)	15	23.930	41.155

Table 2. Moving time and CPU/RAM usage

	Moving time to destination (sec)	**CPU usage (%)**	**Memory usage (MB)**
Off-loading	30.4	12–17	450
On-device	28.8	40–45	1250

Table 3. Camera data latency

Camera position	Latency (end-to-end)
On-public	45.96 ms
On-MEC	27.00 ms
On-device	6.90 ms

- Video resolution: 848 × 480
- Video transmission rate: approximately 4.9 Mbps
- Topic size: 41 KB
- Topics per second: 15 Hz
- JPEG quality: 80

was transmitted at about 2 Hz and the LiDAR at 10 Hz. At this time, the quality of service (QoS) for publishing ROS2 topics was set to "Reliable." This is a way to ensure that the camera sends and receives data more reliably than "Best effort" for simple video viewing [8].

(3) In the case of video, the communication method was to publish the video as a ROS2 topic on the robot and subscribe to it from the video control of the 5G MEC, and the communication between the robot and Kubernetes utilized the cloud bridge described earlier [5]. **Table 3** below summarizes the video topic size, the number of publications per second, the JPEG quality of the video, and the transmission time.

The key challenge in this paper's experimentation was to validate the 5G MEC cloud offloading of the robot's navigation engine in an actual mobile network. Furthermore, we confirmed that various robot-related services, such as control and cluster management, can be utilized in MEC through Kubernetes.

4. Future research directions

Robots are complex integrated systems composed of hardware such as motors, sensors, and mechanical parts, as well as software including operating systems, drivers, communication modules, artificial intelligence, and applications. Therefore, in order to fully leverage the advantages of 5G MEC and cloud robotics, it is necessary to continually modularize and virtualize software through microservice architecture. This will help standardize existing technologies and enable quick integration of new technologies into the robotics field.

Secondly, for fields like artificial intelligence that require abundant resources and automation but not real-time processing, research and development of data collection, storage, and utilization pipelines will be crucial. When applying these to robotics, microservice design should be utilized to reduce entry barriers and increase usability.

Thirdly, while the control of the driving engine in this paper did not require response times of less than 5 seconds, there may be fields that demand real-time robot control at such levels. In such cases, it may not even be possible to attempt cloud robotics adoption and engine offloading. Research on systems with significant time delays has been conducted for a long time, and since the advancement of wireless communication, the field of networked control systems (NCS) [9.10] has seen extensive research. This area needs to investigate possible services and constraints in 5G, which offers ultra-low latency and ultra-reliability.

Lastly, standardization is essential to ensure easy integration of heterogeneous robots and solutions, further advancing technology competitiveness. Current related activities are ongoing, and drafts of "functional requirements" have already been published [11].

Disclosure statement

The authors declare no conflict of interest.

Funding

This project was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (No. 2020-0-00857).

References

[1] Baek SM, Kim YJ, Kim HR, 2020, [The Technical Trend and Research About Intelligent Platform of Cloud Robot]. Robot and Human, 17(3): 3–10. https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE09415746&nodeId=NODE09415746&medaTypeCode=185005&language=ko_KR&hasTopBanner=true

[2] Xia C, Zhang Y, Wang L, et al., 2018, Microservice-Based Cloud Robotics System for Intelligent Space. Robotics and Autonomous Systems, 110: 139–150. https://doi.org/10.1016/ j.robot.2018.10.001

[3] Kubernetes. Viewed Jan 2, 2022. http://kubernetes.io

[4] Kim Y, Lee D, Jeong S, et al., 2021, Development of ROS2-on-Yocto-Based Thin Client Robot for Cloud Robotics. Journal of Korea Robotics Society, 16(4): 327–335. https://doi.org/10.7746/jkros.2021.16.4.327

[5] Woo GH, 2022, The Status of Strategy of LGU+ 5G MEC. HSN2022. Viewed Jan 2, 2022. https://www.hsn.or.kr/board/board.php?task=view&db=board3&no=3&page=1&search=&searchKey=&category=&pageID=ID12160127151

[6] Kang S. cloud_bridge. Viewed Jan 2, 2022. https://github.com/lge-ros2/cloud_bridge

[7] Yang H. lge-ros2/cloisim. Viewed Jan 2, 2022. https://github.com/lge-ros2/cloisim

[8] About Quality of Service Settings. Viewed Mar 2, 2022. https://docs.ros.org/en/rolling/ Concepts/About-Quality-of-Service-Settings.html

[9] Hespanha JP, Naghshtabrizi P, Xu Y, et al., 2007, A Survey of Recent Results in Networked Control Systems. Proceedings of the IEEE, 95(1): 38–162. https://doi.org/10.1109/JPROC.2006.887288

[10] Bigheti JA, Fernandes MM, Godoy EP. 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT), June 4–6, 2019: Control as a Service: A Microservice Approach to Industry 4.0. 2019, Naples, 438–443. https://doi.org/10.1109/METROI4.2019.8792918

[11] Draft Recommendation ITU-T Y.RaaS-reqts: "Cloud computing – Functional requirements for Robotics as a Service". Viewed Mar 2, 2022. https://www.itu.int/md/T17-SG13-210301-TD-WP2-0693.