Journal of Korea Robotics Society (2022) 17(2):118-123
https://doi.org/10.7746/jkros.2022.17.2.118 ISSN: 1975-6291 / eISSN: 2287-3961

5G MEC 기반 로봇 엔진 원격 구동을 위한 클라우드 로보틱스 시스템 구성 및 실증

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

구 세 완[1] · 강 성 규[1] · 정 원 홍[2] · 문 형 일[2] · 양 현 석[2] · 김 영 재†
Sewan Gu[1], Sungkyu Kang[1], Wonhong Jeong[2], Hyungil Moon[2], Hyunseok Yang[2], Youngjae Kim†

Abstract: We implemented a real-time cloud robotics application by offloading 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 5G SA network. 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 are able to avoid obstacles in real-time when the engines are remotely running in 5G MEC server. Test results are compared with 5G Public Cloud and 4G (LTE) Public Cloud as well.

Keywords: 5G, MEC (Mobile Edge Computing), ROS2, Navigation, Cloud, Docker, Kubernetes

1. 서 론

클라우드 로보틱스(Cloud Robotics: CR)를 활용한 다양한 시스템 구성 및 활용과 새로운 시도에 대해서는 그 동안 많은 논문에서 언급이 되어왔다[1,2].

로봇이 네트워크를 통해 클라우드 서버를 활용하는 응용이나 방법은 여러 가지가 있을 수 있다. 클라우드를 통한 로봇 실행 이미지 빌드 / 배포, 로봇 관제, 데이터를 수집 / 적재, AI 활용, 로봇 실시간 제어 등등 이다. 이중 로봇 실시간 제어는 기가 와이파이(WiFi), 5G 통신네트워크 인프라 하에서도 매우 도전적인 과제라 할 수 있다. 그러나 향후 Private 5G 가 활성화 되어 보다 신뢰성(Reliability)과 안정성(Stability)를 갖춘 네트워크로 정착이 된다면, Smart Factory, Smart City, Autonomous Vehicle, Smart Port 등에 충분히 활용 될 수 있다.

본 논문에서는 로봇 엔진(Engine)의 클라우드 오프로딩(Off-loading)을 통하여 로봇 실시간 제어가 가능함을 5G 상용 네트워크 망과 MEC (Mobile Edge Computing) 상에서 실험을 통해 실증하였다.

MEC에서 운영할 클라우드 로보틱스 구성의 플랫폼으로는 쿠버네티스를 채택하였다[2]. 쿠버네티스는 최근 가장 각광 받는 도커(Docker) 오케스트레이션(Orchestration) 플랫폼으로서, 사용자 증가에 따른 전송량(Traffic)에 대처하기 위한 컴퓨터 노드(Node)나 프로세스의 확장성(Scalability), 부하 분산 처리(Load-balancing), 무정지 기능 업그레이드(Rolling-update)등에 강력한 기능들을 갖추고 있을 뿐만 아니라, 특정 도메인에서 서비스 별 요구 사항에 따라 사용자 정의(Customization) 기능을 통해 각 목적에 맞도록 동작을 제어하는 것도 가능하다[3].

5G MEC 서버에 로봇 엔진 중 하나인 자율 주행 엔진(Navigation Engine)을 배포/운영하여 로봇 내부의 온-보드(On-Board)에서 탑재되어 운영되는 것과 같은 성능의 서비스를 구현할 수 있

Received : Feb. 25. 2022; Revised : Mar. 22. 2022; Accepted : Mar. 22. 2022
※ This project was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (No. 2020-0-00857)
1. Chief Researcher, LG Electronics, Seoul, Korea (sewan.gu, sungkyu.kang @lge.com)
2. Senior Researcher, LG Electronics, Seoul, Korea (wonhong.jeong, sungkyu.kang, hyunseok7.yang@lge.com)
† Research Fellow, Corresponding author: LG Electronics, Seoul, Korea (yjae.kim@lge.com)

음을 검증하였고, 로봇 온-보드에서 주행 엔진을 구동 시킬 필요가 없어지기 때문에 중앙 처리 장치(CPU)와 램(RAM) 등 주요 연산 리소스 사용량을 현저히 줄일 수 있을 것으로 판단한다.

본 논문은 총 4개 장으로 구성하였다. 2장에 전체 시스템 구성에 대한 설명을 하였으며, 3장에 실증 및 결과를, 4장에서 향후 연구방향에 대해 기술하였다.

2. 시스템 구성

실증에 사용된 시스템은 크게 3가지 부분으로 나눌 수 있으며, 로봇, 5G 네트워크, MEC 등 이다. 로봇에서 구동되는 운영체제와 플랫폼은 욕토(Yocto)[4]와 ROS2 (Robot Operating System 2)를 활용하였고, MEC에서 구성되는 모든 로봇 엔진은 도커화(Dockerization)하여 선 검증을 진행하였다.

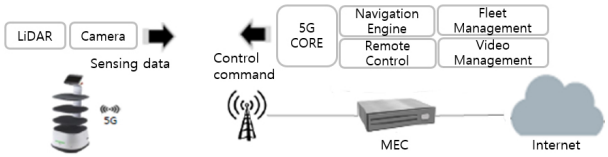
5G 모뎀은 LG전자에서 자체 제작하여 이미 여러 응용 분야에서 사용 중인 검증된 모델을 활용하였다.

다음 2.1, 2.2에서 통신 네트워크 구성 및 클라우드 서비스를 위한 구현 방법에 대해 설명하겠다.

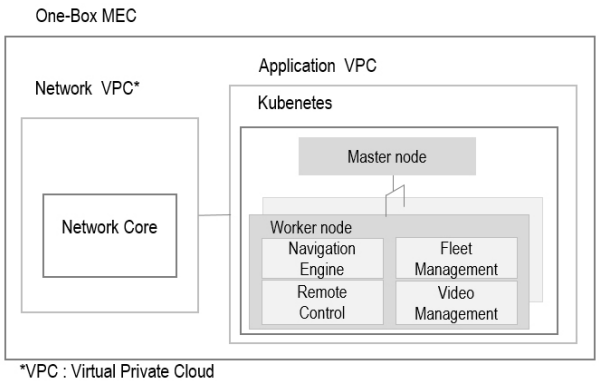
2.1 네트워크 구성

본 논문에서는 U+의 Private 5G MEC에서 실험하였으며, 백본은 실험망이 아닌 상용망을 그대로 활용하였다. [Fig. 1]과 같이 구성이 된다.

클라우드 서버 쪽은 [Fig. 2]와 같이 One-Box MEC 아키텍처를 도입하여 운영되고 있으며, 코어(Core)망 기능을 탑재한



[Fig. 1] Network Architecture



[Fig. 2] One-Box MEC Architecture

클러스터와 응용 프로그램을 서비스하는 클러스터가 있다. 클러스터는 쿠버네티스 클러스터로 마스터 노드가 전체 클러스터를 제어하고 워커노드가 실제 응용 서비스들을 기동하고 운영하는 역할을 담당한다. MEC 에서의 로봇 엔진 구현에 대한 내용은 다음 장에서 다루도록 한다.

실증에 사용한 망은 상용 네트워크를 이용하였으나, MEC는 실증을 위해 일정기간만 활성화되었음을 밝힌다. 해당 내용은 [5]에도 제안 발표되었다.

2.2 클라우드 서비스 구현

클라우드 서비스를 위해서 오픈 표준화 되어 있는 가상화 기술을 활용하며, 이미 언급한 대로 이중 최근 가장 많이 사용하는 기술인 도커 / 컨테이너 기술을 활용 하였다. 복잡한 구조를 손쉽게 분산 배치하고 운용할 수 있는 마이크로 서비스 아키텍처에 적합하다.

어떤 서비스나 응용 프로그램도 도커 이미지로 만들어 놓으면, 도커 엔진이 구동되는 호스트 컴퓨터(Host Computer)에서 실행할 수 있기 때문에 분산 환경과 관리에 매우 편리한 기술이다. 또한 하드웨어, 운영체제로 부터 매우 독립적으로 운영할 수 있어서 빠르게 배포 서비스할 수 있는 장점도 있다.

따라서 로봇에서 실행되는 모든 단위의 기술들을 최소 단위로 모듈화하고, 도커 이미지로 생성하게 되면, 실제 물리적인 위치와 독립적으로 실행시키고, 상호 동작 시킬 수 있는 아키텍처를 구현할 수 있는 것이다.

본 연구에서는 MEC 쿠버네티스에 로봇 관련 엔진을 4개 구동시켰다. 주요 기능인 주행 엔진, 동일한 장소에서 멀티 로봇

```
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

[Fig. 3] Desploy yaml file for Kubernetes

경로 계획을 위한 군집 제어(Fleet Management), 원격에서 로봇을 제어하고 위치를 감시하고 상태를 검사하는 원격 제어(Remote Control), 로봇의 전방 카메라로부터 획득한 동영상 이미지를 출력하는 영상 관제(Video Management)로 구성된다.

또한 이러한 엔진들이 외부 로봇과 통신하기 위해서는 쿠버네티스 네트워크 설정에서 외부로의 네트워킹 이 가능하도록 해야 하고, 이를 위해 NodePort를 외부 네트워크와 통신을 위한 포트로 설정하여 구현하였다. NodePort는 외부에서 특정 응용 서비스에 접근시에 쿠버네티스 내의 어떤 클러스터 Node에서 실행이 되든 동일한 Port를 통해 접근할 수 있도록 해주는 쿠버네티스에서 제공하는 서비스이다. 주행 엔진을 배포/ 설치하는 방법과 함께 이에 대한 내용 일부가 [Fig. 3]에 예시되어 있다.

3. 실증 및 결과

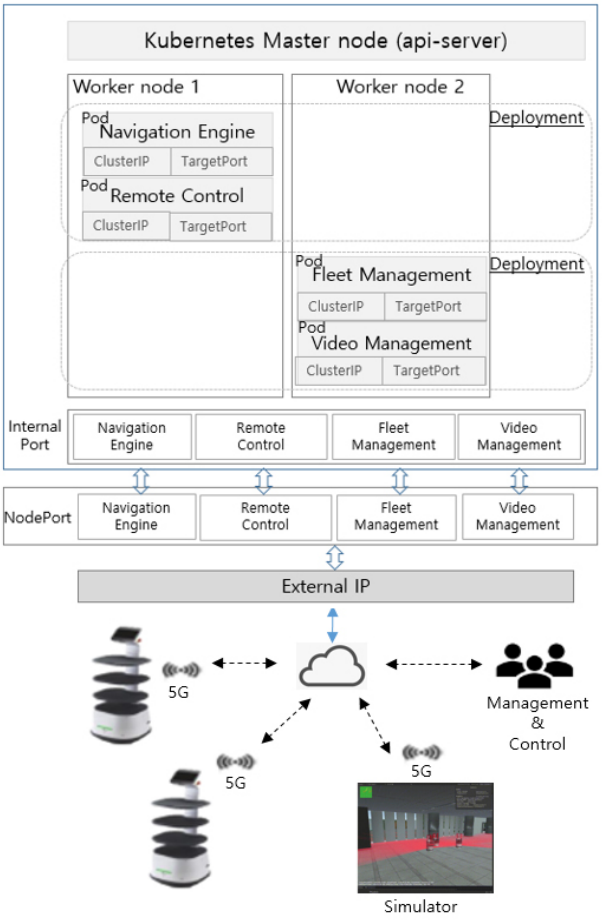
실험을 위한 로봇은 레스토랑에서 활용할 수 있는 자율주행 배송로봇을 활용하였다. 모터를 구동하는 구동부와 구동부에 명령을 주는 주요 어플리케이션 연산을 위한 하드웨어는 Nvidia사의 TX2 이며, 플랫폼은 2장에서 언급한 경량 운영체제인 Yocto기반의 리눅스 운영 체제와 ROS2 프레임 워크를 사용했다[4]. ROS2 프레임워크는 로봇 어플리케이션간에 다양한 센서, 응용 메시지 정보 등등을 손쉽게 송수신해주는 플랫폼으로 다양한 로봇 개발 툴들도 제공이 되고 지속 발전되고 있어서 로봇 개발자들에게는 각광을 받고 있다.

본 논문 실증에서 가장 중요한 어플리케이션은 자율 주행 엔진이며, 로봇으로부터 라이다(LiDAR), 카메라 센서 데이터들을 5G망을 통해서 수신하여, 정해진 지도(Map)를 기반으로 주어진 이동 목적지에 맞추어 주행 경로를 결정하는 엔진이다. 따라서 주행 엔진을 모듈화하고, 도커화하여 클라우드 서비스를 가능하도록 2장에서 설명한 것과 같이 준비 하였다.

추가적으로 5G MEC 서버와 로봇이 위치한 지점은 서브네트워크가 달라서 ROS2 메시지를 주고 받을 수 없다. 이러한 문제를 극복하기 위해 로봇과 서버간의 ROS2 메시지를 마치 같은 서브네트워크에 있는 것처럼 송수신할 수 있도록 클라우드 브리지[6]라는 개발된 모듈을 활용하였는데, 이는 서로 다른 네트워크에 있는 호스트에서 ROS2 메시지 통신을 가능하게 하는 통신 플랫폼으로 5G 네트워크에서 다양한 외부 호스트와의 ROS2 응용 프로그램 간의 통신에 적합하다고 할 수 있다.

사업자 MEC내의 쿠버네티스 클러스터에서 실행되는 클라우드 로보틱스 시스템 구성도가 [Fig. 4]에 예시되어 있다.

실험은 양재동에 준비한 테스트베드(Testbed)에서 U+ 5G 실증망을 통해 일정 기간 수행하였다.



[Fig. 4] Cloud Robotics system on 5G MEC Kubernetes

3.1 실증 실험 방법

5G 운영망에서의 실증을 하기에 앞서 5G MEC를 갖춘 통신 사업자 테스트베드에서 우선적으로 검증 진행을 하였다. 이때 로봇 대신 로봇의 물리적인 센서 및 구동을 모사하고, 공간도 3D 로 모델링한 시뮬레이터를 활용하였다[7]. 라이다 등 센서 데이터, 모터 구동 데이터를 서버에 보내고, 서버에서는 로봇의 주행 엔진을 통해 계산된 주행 명령을 내려서 시뮬레이터에서 구동되는 로봇이 주행할 수 있도록 하였다. 기본적인 주행에 문제가 없음을 검증 확인하여, 이후 실제 운영망에서 진행할 수 있었다.

테스트베드를 레스토랑으로 가정하였으며, N 개의 고객 테이블을 M 대의 자율주행 로봇이 배송 서비스를 하는 시나리오였다. 이때 각 로봇의 제어를 담당하는 주행 엔진 이외에 스케줄링은 군집 제어를 통해서 이루어지며, 영상 관제를 통해 배송 로봇에 장착된 카메라를 통해 바라보는 실시간 동영상을 확인할 수 있다. 원격제어 모듈을 통해 로봇 위치를 확인하고 이상 상태를 감지 혹은 수동 명령을 내릴 수 있도록 하였다. 주

행엔진은 로봇당 하나가 할당되어 실행되며, 기타 엔진은 로봇의 대수와 무관하게 하나의 엔진이 구동 실행된다.

3.2 실증 결과

로봇과 5G 코어망까지 패킷 전송구간은 로봇(단말) -> RAN (Radio Access Network) -> UPF(User Plane Function) -> MEC 상의 로봇 엔진(마이크로서비스)까지의 구간이다. 이 구간에서 5G, 4G의 기본적인 지연시간(latency)를 ICMP 프로토콜의 ping 테스트에 의해 먼저 평균치를 분석해봤으며, [Table 1]에 수치화 하였다. 해당 데이터는 동영상과 같은 많은 양의 데이터를 주고 받을 때 필요한 대역폭을 고려하지 않은 단순한 ping 에서 사용하는 56바이트로 구성된 패킷(Packet)에 대한 것이다.

1) 예상대로 5G core 망이 4G (LTE) 대비 약 2.8배 가까이 빠른 것을 확인하였고, 5G Public Cloud보다 MEC에서 1.6배 가량 빠른 것을 검증하였다.
2) 1시간 가량의 2대 로봇 3군데 목적지 주행을 스케줄링을 통해 수행했을 때 문제없이 온-보드와 동일한 성능으로 자율주행을 했으며, 로봇 시스템에서의 중앙 처리 장치와 램 메모리의 사용량을 비교해본 결과 [Table 2]와 같이 기존 온-보드 시스템 대비 약 50% 이상 효율이 있음을 확인했다. 자율 주행에 활용된 로봇의 센서 데이터는 카메라와 라이다 이며, 카메라는 약 2 Hz, 라이다는 10 Hz로 발행 전송하였다. 이때 ROS2 토픽 발행을 위한 QoS 는 “Reliable” 로 설정하였다. 단순 영상을 보기 위한 카메라가 “Best effort” 보다 더 확실하게 데이터를 송수신을 확인하는 방법이다[8].
3) 동영상의 경우 로봇에서 영상을 ROS2 토픽(Topic)으로 발행(Publish)하고 5G MEC의 영상 관제에서 구독(Subscribe)하여 받는 통신 방식이었으며, 로봇과 쿠버네티스 사이 통신은 앞에서 설명한 클라우드 브리지를 활용하였다[5]. 아래 [Table 3]에 동영상 토픽 크기, 초당 발행 수, 동영상을 구성하는 JPEG 품질 및 이에 대한 전송시간을 정리하였다.

[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	27.0 ms
On-Device	6.9 ms

- 동영상 해상도 : 848 × 480
- 동영상 전송 속도 : 약 4.9 Mbps
- 토픽 크기 : 41 KB
- 초당 토픽 발행 수 : 15 Hz
- JPEG quality: 80

본 논문에서 주요 실증 과제는 클라우드 로보틱스 기술 중 로봇 주행 엔진의 5G MEC 클라우드 오프로딩을 실제 이동통신 사업자 망에서 실행하여 5G MEC에서 자율 주행 서비스에 대한 가능성을 검증하였다. 이외 관제, 군집제어 등 다양한 로봇 관련 서비스를 MEC 상에서 쿠버네티스를 통해 활용될 수 있음을 확인하였다.

4. 향후 연구 방향

로봇은 모터 / 센서 / 기구부 등 하드웨어와 더불어 운영체제 / 드라이버 / 통신 모듈 / 인공지능 / 응용 프로그램으로 구성된 소프트웨어의 복잡한 통합 시스템이라 할 수 있다. 따라서, 첫 번째, 5G MEC 및 클라우드 로보틱스를 통한 충분한 장점 활용을 위해서는 소프트웨어의 모듈화 및 가상화 기술을 통해 지속적으로 마이크로서비스(Microservice)화를 진행해야 한다. 이를 통해 이미 알려져 있는 기술들을 보편화하고, 새로운 기술들을 빠르게 로봇 분야에 접목할 수 있도록 할 수 있는 토대를 마련할 수 있다.

두 번째, 실시간성이 필요치 않으나, 많은 자원과 자동화가 필요한 인공지능과 같은 분야의 클라우드 컴퓨팅 활용을 위해서는 데이터를 수집, 적재, 활용하는 파이프 라인(Pipeline)이 중요한 연구 개발 분야가 될 것이고, 이것도 로봇에 적용 시 마이크로서비스 설계를 활용하여 진입 장벽 및 활용도를 높여야 한다.

세 번째, 본 논문에서의 주행 엔진에 대한 제어는 5 ms 이하 단위에서 응답 속도를 보장해야 하는 정도의 응용은 아니었으나, 이러한 정도의 실시간으로 로봇을 제어해야 분야가 있다. 어쩌면 이런 분야는 해당 엔진의 클라우드 로보틱스 도입 및 엔진의 오프로딩 시도조차 할 수 없었을지 모른다. 오래 전부터 시간 지연을 갖는 시스템에 대해 연구가 있었으며, 유무선 통신의 발달 이후로는 주로 Networked Control System (NCS)

[9,10] 이라는 분야로 많은 연구가 되어 오고 있다. 이를 통해 초지연, 초연결성을 갖는 5G 에서의 가능한 서비스와 제약에 대한 연구가 이루어져야 할 것이다

마지막으로 표준화를 통해 이종 로봇과 솔루션들이 손쉽게 통합되어 더욱 경쟁력 있는 기술로 발전시켜야 할 것이다. 현재 관련된 활동이 진행 중이며, Functional Requirements 의 Draft 도 발표되었다[11].

References

[1] S. M. Baek, Y. J. Kim, and H. R. Kim, "The Technical trend and Research about intelligent platform of cloud robot," *Robot and Human*, pp. 3-10, July, 2020, [Online], https://www.dbpia.co.kr/pdf/pdfView.do?nodeId=NODE09415746&mark=0&useDate=&ipRange=N&accessgl=Y&language=ko_KR&hasTopBanner=true.

[2] C. Xia, Y. Zhang, L. Wang, S. Coleman, and Y. Liu, "Microservice-based cloud robotics system for intelligent space," *Robotics and Autonomous Systems*, pp.139-150, 2018, DOI: 10.1016/j.robot.2018.10.001.

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

[4] Y. Kim, D. Lee, S. Jeong, H. Moon, C. Yu, K. Lee, J. Choi, and Y. Kim, "Development of ROS2-on-Yocto-based Thin Client Robot for Cloud Robotics," *Journal of Korea Robotics Society*, pp. 327-335, Dec., 2021, DOI: 10.7746/jkros.2021.16.4.327.

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

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

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

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

[9] J. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survery of recent results in networked control systems," *IEEE*, vol. 95, no. 1, pp, 38-162, 2007, DOI: 10.1109/JPROC.2006.887288.

[10] J. A. Bigheti, M. M. Fernandes, and E. P. Godoy, "Cotrol as a Service: A Microservice Approach to Industry 4.0," *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT)*, Naples, Italy, 2019, DOI: 10.1109/METROI4.2019.8792918.

[11] Draft Recommendation ITU-T Y.RaaS-reqts: "Cloud computing - Functional requirements for Robotics as a Service," *ITU-T SG 13 (Study Period 2017) Temporary Document 693-WP2*, 2021, [Online], https://www.itu.int/md/T17-SG13-210301-TD-WP2-0693.

구 세 완

1994 한양대학교 전자통신공학(학사)
1996 한양대학교 전자통신공학(석사)
2013 한양대학교 전자컴퓨터통신공학 (박사수료)
1997~2000 한국철도기술연구원 주임연구원
2000~현재 LG전자 책임연구원

관심분야: Cloud Robotics, Robot Platform, Networked Control System, Robust Control

강 성 규

2001~2008 서울시립대학교 전자전기컴퓨터공학부(학사)
2008~2010 서울시립대학교 전자전기컴퓨터공학부(석사)
2010~2014 한국과학기술연구원 지능로봇사업단 연구원
2014~2019 로보케어 책임연구원
2019~현재 LG전자 책임연구원

관심분야: Cloud Robotics, Robot Platform, Fleet Management System

정 원 홍

2013 한양대학교 전자통신공학과(학사)
2016 한양대학교 전자통신컴퓨터공학부 (석사)
2016~현재 LG전자 선임연구원

관심분야: Cloud Robotics, Robot Platform, Kubernetes, Big Data

문 형 일

2013 부산대학교 컴퓨터공학부(학사)
2014~현재 LG전자 선임연구원

관심분야: Embedded System, Robot Operating System, SLAM

양 현 석

2010 숭실대학교 정보통신전자공학부(학사)
2012 고려대학교 컴퓨터학(석사)
2012~2018 Ericsson-LG, 선임연구원
2018~2021 LG전자, 선임연구원
2022~2022 LG전자, 책임연구원

관심분야: Cloud Robotics, Robotics Simulation, Robot Platform,Multi-Agent Path Finding

김 영 재

1999 서울대학교 전기공학부(학사)
2003 Stanford Univ. 전기공학(석사)
2007 Stanford Univ. 전기공학(박사)
2009~2017 Apple Inc., 시니어 SW엔지니어
2017~2018 Velodyne LiDAR, 수석엔지니어
2019~현재 LG전자 연구위원

관심분야: Cloud Robotics, Network, Robot Operating System