electronics

MDPI

Article

Enhancing Autonomous Driving Robot Systems with Edge Computing and LDM Platforms

Jeongmin Moon, Dongwon Hong, Jungseok Kim, Suhong Kim, Soomin Woo, Hyeongju Choi and Changjoo Moon *

Department of Smart Vehicle Engineering, Konkuk University, Seoul 05029, Republic of Korea; mjm9804@konkuk.ac.kr (J.M.); dks01972@konkuk.ac.kr (D.H.); dyori04@konkuk.ac.kr (J.K.); britzes@konkuk.ac.kr (S.K.); soominwoo@konkuk.ac.kr (S.W.); hj8709@konkuk.ac.kr (H.C.)
* Correspondence: cjmoon@konkuk.ac.kr

Abstract: The efficient operation and interaction of autonomous robots play crucial roles in various fields, e.g., security, environmental monitoring, and disaster response. For these purposes, processing large volumes of sensor data and sharing data between robots is essential; however, processing such large data in an on-device environment for robots results in substantial computational resource demands, causing high battery consumption and heat issues. Thus, this study addresses challenges related to processing large volumes of sensor data and the lack of dynamic object information sharing among autonomous robots and other mobility systems. To this end, we propose an Edge-Driving Robotics Platform (EDRP) and a Local Dynamic Map Platform (LDMP) based on 5G mobile edge computing and Kubernetes. The proposed EDRP implements the functions of autonomous robots based on a microservice architecture and offloads these functions to an edge cloud computing environment. The LDMP collects and shares information about dynamic objects based on the ETSI TR 103 324 standard, ensuring cooperation among robots in a cluster and compatibility with various Cooperative-Intelligent Transport System (C-ITS) components. The feasibility of operating a large-scale autonomous robot offloading system was verified in experimental scenarios involving robot autonomy, dynamic object collection, and distribution by integrating real-world robots with an edge computing–based offloading platform. Experimental results confirmed the potential of dynamic object collection and dynamic object information sharing with C-ITS environment components based on LDMP.

Keywords: 5G MEC; driving robots; offloading; Kubernetes; C-ITS

Citation: Moon, J.; Hong, D.; Kim, J.; Kim, S.; Woo, S.; Choi, H.; Moon, C. Enhancing Autonomous Driving Robot Systems with Edge Computing and LDM Platforms. *Electronics* **2024**, *13*, 2740. https://doi.org/10.3390/electronics13142740

Academic Editor: Mahmut Reyhanoglu

Received: 18 June 2024
Revised: 4 July 2024
Accepted: 10 July 2024
Published: 12 July 2024

1. Introduction

The rapid advancement of autonomous robots is positioned at the forefront of modern technology; thus, the study of efficient robot operations and inter-robot interoperability is becoming increasingly important. Research on the operation of robots, particularly through advanced functionalities, e.g., swarm control, continues to evolve to enable their roles in a wide range of applications, including security, environmental monitoring, and disaster response. To ensure continuous environmental awareness and efficient operation of autonomous robots designed for diverse applications, processing large volumes of sensor-generated data is essential. Computations based on large volumes of data can enhance the independent operability of autonomous robots significantly; however, these computations also impose considerable computational burdens, which lead to battery consumption and heat issues. For example, with autonomous robots, this computational burden affects operational time and driving distance significantly.

In addition, the embedded environment configuration of autonomous robots must satisfy specific requirements, e.g., real-time processing, high durability, and robustness against shock and vibration, and these characteristics incur significantly higher setup and operational costs compared to general computing environments.

To address these challenges, offloading robot computations to edge computing is gaining attention. Edge computing is a computing paradigm where data processing and analysis are performed close to the data source rather than in a remote centralized data center. As shown in Table 1 for edge-based offloading, it is designed to minimize the network bandwidth usage and reduce the response time. Therefore, offloading the computations of autonomous robots to edge computing can considerably enhance their responsiveness. Additionally, compared with the onboard processing method, it can effectively reduce the power consumption of robots and setup costs of autonomous driving systems [1]. In addition, offloading data related to dynamic objects in the surroundings of autonomous robots improves the ability to share information about such dynamic objects with other robots in the same swarm or with other mobility systems in real-time. This real-time sharing of dynamic object information can enable effective collaboration among robots in the swarm, thereby maximizing the efficiency and safety of autonomous driving [2].

Table 1. Comparison with alternative approaches.

Approach	Advantage	Disadvantage
Edge-based offloading	Lower energy consumption Reduced latency due to proximity to the data source Lower bandwidth usage Decreased weight and complexity of the robot Enhanced ability to share information with other devices	Usable only in specific areas where MEC is deployed Dependence on stable internet connectivity
Cloud-based offloading	Lower energy consumption Decreased weight and complexity of the robot Enhanced ability to share information with other devices	Higher Latency Dependence on stable internet connectivity
Onboard processing	Complete autonomy and independence from network Immediate processing without transmission delays	Limited by robot computational power Higher energy consumption Increased weight and complexity of the robot

Thus, this study proposes an autonomous robot operation platform that combines an Edge-Driving Robotics Platform (EDRP) for 5G mobile edge computing (MEC)-based autonomous robot offloading and a Local Dynamic Map Platform (LDMP) for real-time dynamic object information sharing, as shown in Figure 1. The EDRP is designed using a microservice architecture (MSA) to offload the functions of multiple autonomous robots and consider system scalability. These MSA-based autonomous driving modules are deployed in a Kubernetes environment to provide high availability and flexible utilization of computing resources.

The LDMP is designed to analyze the data offloaded from the autonomous robots in real-time and share dynamic object information corresponding to layers 3 and 4 of the LDM concept with C-ITS components, e.g., RSUs and nearby mobility systems. To achieve this, the dynamic object data are converted and distributed using the Collective Perception Message (CPM), as specified in the C-ITS dynamic object message standard ETSI TR 103 324 [3].

In this study, we present and validated a mechanism for data processing and computational offloading of autonomous robots using 5G MEC and Kubernetes. By modularizing the autonomous driving functions and deploying them on the edge cloud, we demonstrated the ease of updating and maintaining individual autonomous driving functions. Finally, we established the LDMP, which can collect information regarding dynamic objects acquired from a multirobot and the autonomous driving offloading platform. This confirmed the feasibility of operating a large-scale autonomous robot offloading system based on LDM

through the collection and sharing of dynamic object information with components in a C-ITS environment.

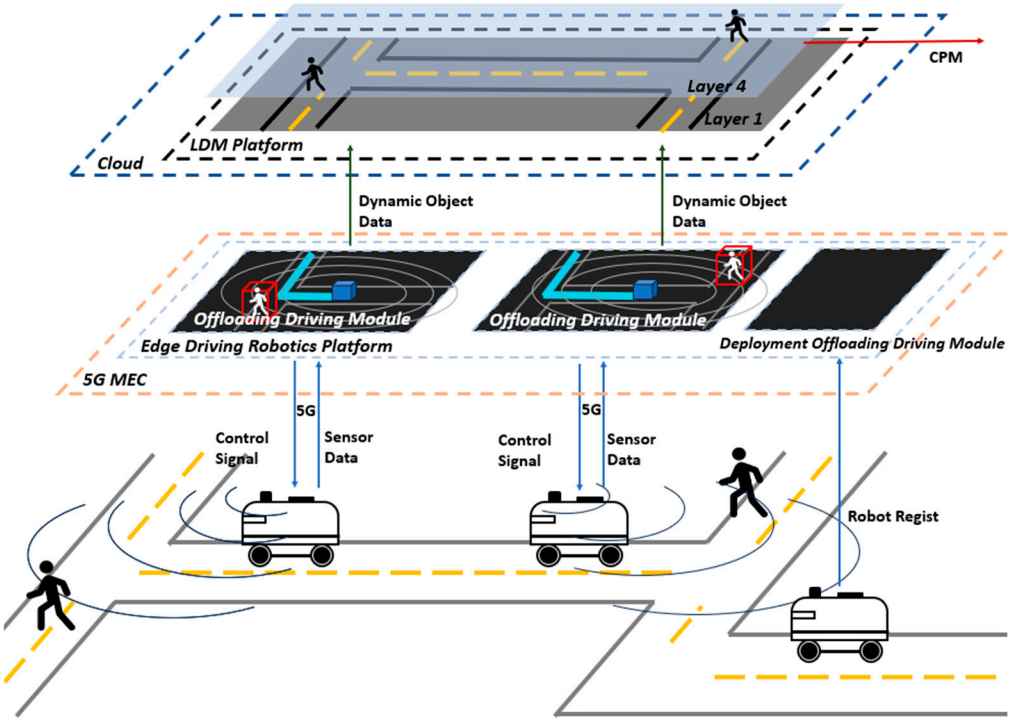

Figure 1. Edge-Driving Robotics Platform (EDRP) and Local Dynamic Map Platform (LDMP).

The primary contributions of this study are summarized as follows.

1. A platform is proposed to enable offloading for autonomous robots and sharing of dynamic object information acquired based on the offloaded data.
2. The platform ensures the operational efficiency and high availability of autonomous robots by designing the offloading modules using an MSA and deploying them in a Kubernetes environment.
3. Offloading overcomes the computing resource limitations of existing autonomous robots, thereby allowing the application of high-performance artificial intelligence (AI) to the autonomous robot environment without requiring computing resource upgrades.
4. By applying the ETSI TS 103 324 standard in the process of sharing information about the dynamic objects acquired during robot operation, the feasibility of applying actual autonomous robots in the C-ITS environment is enhanced [3].

The remainder of this paper is organized as follows. Section 2 introduces relevant background information and related work. Section 3 describes the architectures of the proposed EDRP and LDMP, and Section 4 presents the implementation results and experimental outcomes. Finally, the paper is concluded in Section 5, including a brief discussion of potential future research directions.

2. Background and Related Work

To the best of our knowledge, previous studies have not investigated systems where cloud offloading and the LDMP for robots coexist. Thus, this section provides an overview of previous studies on cloud offloading and dynamic object sharing through the LDMP in autonomous robots. Specifically, Section 2.1 presents relevant background information, and Section 2.2 discusses related work pertinent to the current study.

2.1. Background

The technologies that facilitate EDRP and LDMP are presented below.

(1) 5G MEC

In the 5G network architecture, MEC introduces cloud computing capabilities at the network edge, which enables real-time, high-bandwidth, and low-latency access to wireless network information [4]. This technology enhances the efficiency of data processing, reduces network traffic, and improves service speed and reliability.

(2) Robot Operating System

The Robot Operating System (ROS) is an integrated middleware designed for robot development that focuses on providing standard interfaces for various hardware and algorithms in line with the advancement in robotic technology [5,6]. In addition, the node-based architecture of the ROS facilitates effective communication and collaboration between sensors, actuators, and algorithms [7].

(3) Kubernetes

Kubernetes is an open-source platform that automates the deployment, scaling, and management of containerized applications [8,9]. Originally designed by Google to optimize application deployment in cloud environments, Kubernetes is now managed by the Cloud Native Computing Foundation. It automates the deployment and operation of containers across the nodes in a cluster and offers a wide range of core functionalities, e.g., service discovery, load balancing, storage orchestration, self-healing, and scaling [10].

(4) Microservices Architecture

The MSA is a design approach that divides a large application into multiple small, independently deployable services [11]. Each microservice performs a specific function and can be developed, deployed, and scaled independently. Compared to a monolithic architecture, this structure realizes easier updates and scalability because each service can be deployed and operated independently [12].

(5) Message Queuing Telemetry Transport

The Message Queuing Telemetry Transport (MQTT) protocol is designed to operate efficiently on low-power, resource-constrained devices, e.g., Internet of Things devices [13]. The MQTT protocol utilizes a publish/subscribe model, where messages are published to or subscribed from topics via a central server (referred to as a broker).

(6) Collective Perception Message

As specified in the ETSI TR 103 324 standard, The CPM is used by Intelligent Transport Systems (ITS) stations to share information about perceived objects and perception areas [3,14]. For example, the CPM facilitates data fusion between ITS stations by providing information about vehicles and other objects on the road. In addition, the CPM includes the observation status and attributes of perceived objects, e.g., detection time and position, and may include other elements as required, e.g., motion and posture status.

(7) Local Dynamic Map

The LDM technology is used to store static and dynamic information collected from vehicles and infrastructure, which are used to visualize the characteristics and status of the surrounding environment [15]. The LDM includes information across multiple layers, i.e., layer 1 (permanent static data, including high-precision electronic maps), layer 2 (temporary static data, e.g., road infrastructure and traffic signs), layer 3 (temporary dynamic data, including traffic information and local weather data), and layer 4 (dynamic data involving vehicles and pedestrians, and other data relevant to autonomous driving).

2.2. Related Work

This section summarizes previous studies on cloud offloading and C-ITS in autonomous robots. Research on computational offloading for robots has evolved continuously. For example, Yuanzhao Zhai highlighted the importance of computational offloading in the field of driving robots and proposed a collaborative offloading approach among

robots [16]. In addition, Wang et al. presented a 5G MEC-based intelligent computational offloading framework for power robot inspection to address multidimensional object heterogeneity, environmental dynamics, and inspection delay guarantees [17]. Liu proposed the novel RoboEC2 cloud robotic system, which supports dynamic network offloading for driving robots [18]. Zhang proposed a UAV-assisted task-offloading system that employs the dung beetle optimization algorithm and deep reinforcement learning to minimize latency and energy consumption in disaster rescue scenarios [19]. Sossalla investigated offloading the core functions of mobile robots, e.g., simultaneous localization and mapping, to the edge cloud, significantly reducing the computational requirements and energy consumption of robots [20]. In addition, de Souza examined how a container-based edge offloading framework can support autonomous driving [21].

Research is actively ongoing on efficient data transmission and processing methods centered on edge computing, cloud computing, and the 5G and 6G network technologies, which are fundamental for cloud robotics. For example, Ho proposed an optimized task-offloading scheme for UAV-assisted cloud robotics [22]. This scheme utilizes Karush–Kuhn–Tucker conditions, Lagrangian dual decomposition, and the Proximal Policy Optimization method for offloading decision making and computational resource allocation. In addition, Wei developed a dual-agent reinforcement learning algorithm to optimize task offloading and mobility control in hybrid satellite–terrestrial networks, thereby reducing offloading delays in mobile robots [23]. Tang proposed a concept to overcome the high cost of onboard sensors in autonomous vehicles by installing LiDAR sensors on roadside lamp posts to share data with passing vehicles [24]. Kong confirmed that using MEC technology can reduce the latency and improve throughput for 5G multimedia and vehicle-to-everything (V2X) applications [25]. Srinivasa designed and evaluated a remote driving control system that supports real-time video streaming over wireless networks [26]. Yu and Lee discussed the issue of onboard computing resource shortages and suggested that edge computing technology and 6G networks could provide solutions to this problem [27]. In addition, Wang et al. demonstrated through experiments involving a local edge computing environment that MEC can address the latency and reliability constraints of centralized cloud computing, thereby enabling specific autonomous and assisted driving tasks [17]. Coronado proposed a LiDAR-based real-time streaming framework with data collection, compression, encryption, and encoding from LiDAR sensors followed by data transmission via WiFi [28]. Ayoub and Villing investigated the scalability of computation offloading for robots over WiFi networks [29], and Zhu proposed a reinforcement learning-based multi-agent task-offloading and network selection scheme to improve the QoS and energy efficiency of smart farms [30].

Many researchers studying C-ITS are focusing on the sharing of dynamic objects on roads using cloud and edge computing. For example, Shin used edge-fog-cloud computing to share dynamic objects in real-time [31]. Yoo developed a real-time big data analysis system based on the Hadoop ecosystem to detect and share objects on the road [32]. Cho proposed a projection method from local to global coordinate systems using traffic surveillance cameras and developed a platform to distribute information about dynamic objects in real-time via Kafka [33].

The current literature has investigated and utilized the computational offloading of autonomous robots and C-ITS-based dynamic object collection systems in separate environments. However, to the best of our knowledge, no study has explored the integration of these technologies. The combination of cloud offloading for autonomous robots and the LDMP enables the real-time sharing of object information on roads, thereby creating a safer operational environment for robots. In addition, this integration maximizes the compatibility with C-ITS in complex urban environments, thereby improving the operation and utility of autonomous robots, as well as providing a safer operational environment through the interaction of autonomous robots and vehicles.

This paper proposes a solution that combines the EDRP, which ensures high availability and efficient utilization of computing resources in the computational offloading of

autonomous robots, and the LDMP, which enhances the robots' environmental awareness and interaction capabilities through dynamic object collection. This integrated platform offers a new direction to maximize the advantages of integrating autonomous robots with the C-ITS environment, including high availability in robot offloading and efficient utilization of computing resources.

3. EDRP and LDM Platform Architecture

Figure 2 shows the modules comprising the proposed platform. The computational process of the offloaded autonomous driving of robots is performed by 5G MEC. Robots are equipped with a sensor system including LiDAR and GPS, and a 5G module, all of which are based on the ROS middleware. Communication between the robots and the 5G MEC is established through a C-V2X Uu interface.

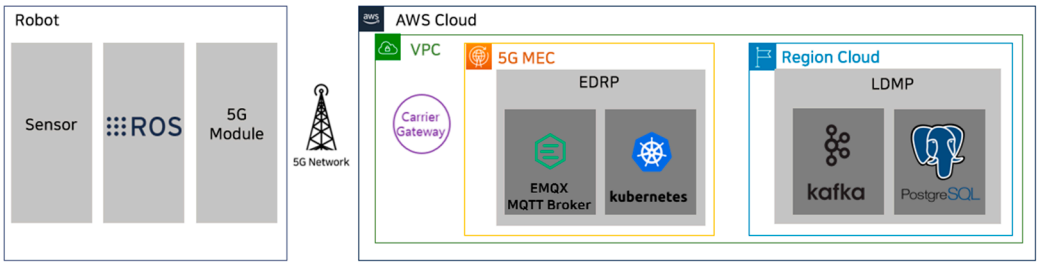

Figure 2. EDRP and LDMP environments.

The EDRP involves an EMQX MQTT broker and a Kubernetes environment. Here, the EMQX MQTT broker is employed to transmit the sensor data and control messages between robots. The proposed platform adopts a centralized message broker to ensure secure communication among multiple robots and ROS environments configured in different settings, and offloading involves containerizing each autonomous driving function. These containerized robot autonomous driving functions are deployed and operated in a Kubernetes environment.

Note that the LDMP does not require ultralow latency; thus, it is deployed in a region cloud environment rather than 5G MEC. The LDMP comprises both Kafka and PostgreSQL. Here, Kafka is employed to transmit dynamic object information obtained from the EDRP to the LDMP and to other C-ITS components, e.g., RSUs and surrounding vehicles. The PostgreSQL database, enhanced with the PostGIS spatial information extension, is employed to store dynamic object information.

Figure 3 illustrates the data flow in the proposed platform. First, the GPS and LiDAR are published through the MQTT broker built on 5G MEC. This information is received by the EDRP, which is deployed on AWS Wavelength as 5G MEC. The EDRP receives each sensor's data via MQTT. Based on the received data, it processes the information using localization and object recognition algorithms to perform the offloaded computational tasks for autonomous driving.

The global planner receives the target driving destination from the user, processes this information, and generates a route to the destination using the shortest path. Then, the local planner generates control signals for the driving robot and transmits them via MQTT. MQTT provides network reliability by transmitting and receiving sensor data using TCP-based Transport Layer Security (TLS).

The LDMP receives the dynamic object information detected during perception via Kafka. Here, the LDM bridge converts this information into CPM format as per the ETSI TS 103 324 standard. The converted data are then stored in the LDMP and distributed to other C-ITS components via Kafka. Furthermore, Kafka offers network reliability by transmitting and receiving data using TCP-based TLS.

In the following sections, we provide a detailed overview of the robot, EDRP, and LDMP, which comprise our proposed framework.

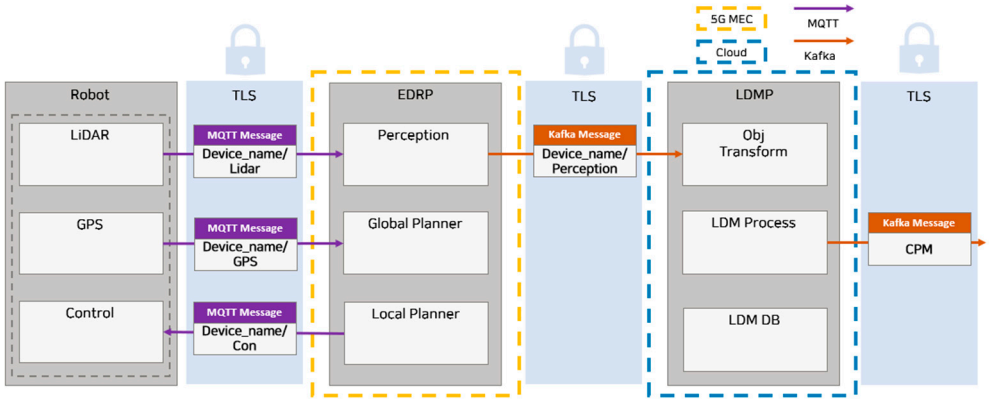

Figure 3. Data flow in the proposed platform.

3.1. Robot

3.1.1. Robot System

As shown in Figure 4, the device's sensor configuration includes LiDAR, GPS, and a 5G module. The robot is connected to the EDRP via V2N communication through the 5G Uu interface.

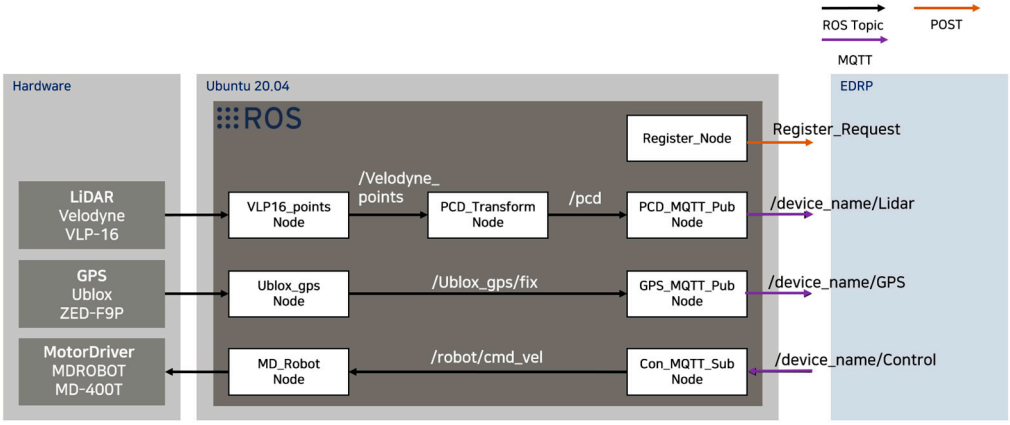

Figure 4. Autonomous robot architecture and data flow.

Table 2 shows messages used by autonomous robots. Prior to offloading to the EDRP, the robot sends a postmessage to the EDRP through Register_Request, as specified in Table 2. After registration, the connected robot publishes its location information 10 times per second from the attached GPS sensor via the Ublox_gps node to the ROS topic /ublox_gps/fix. These published GPS data are then transmitted by the GPS_MQTT_Pub node to the /device_name/GPS topic. For LiDAR, the data are published through the Velodyne node to the /Velodyne_points topic. The Poincloud2_to_pcd node converts the point cloud data to the PCD compression format to facilitate efficient transmission. The converted PCD compression data are then sent by the PCD_MQTT_PUB node to the /device_name/Lidar message topic. Regarding robot control, the Con_MQTT_SUB node receives control messages from the /device_name/control topic, and these messages are then transmitted to /robot/cmd_vel. The md_robot node receives these messages, converts them to the RS-485 communication system, and controls the motors through the motor driver.

Table 2. Message information in autonomous robot.

Name	Protocol	Format	Description
Register_Request	HTTP	POST Request	Registers a robot with a EDRP
/ublox_gps/fix	ROS Topic	Sensor_msgs/NavSatFix	GPS data (including latitude and longitude)
/Velodyne_points	ROS Topic	Sensor_msgs/PointCloud2	LiDAR point cloud data from Velodyne sensor
/robot/cmd_vel	ROS Topic	Geometry_msgs/Twist	Commands for device's linear and angular velocities
/device_name/Lidar	MQTT	Custom JSON	LiDAR PCD compression data from Velodyne sensor
/device_name/GPS	MQTT	Custom JSON	GPS data from Ublox GPS sensor
/device_name/control	MQTT	Custom JSON	Control commands to the device

3.1.2. Robot Risk-Management Mechanism

Table 3 describes the Adaptive Data Transmission Frequency Control (ADTFC) algorithm, which adjusts the data transmission frequency (Hz) to handle unstable network conditions, e.g., latency spikes during data communication between the robot and EDRP. The ADTFC algorithm is inspired by the AIMD Sawtooth function used in TCP Congestion Control to regulate the data transmission frequency. Here, the robot measures the network status using the EDRP based on the Control Message Subscribe time interval presented in Table 1. In addition, the initial and maximum data transmission frequencies of the robot were set to 10 Hz, which is the data publish rate of /Ublox_gps/fix provided in Table 2. Based on the maximum human reaction time, the threshold for the Message Subscribe Time Interval was set to 150 ms. If this interval exceeds 150 ms, the data transmission frequency decreases; otherwise, the transmission frequency increases.

Table 3. Pseudo code of the adaptive transmission frequency control.

```
// Definitions
initial_freq: Initial Frequency (10 Hz)
max_freq: Maximum Frequency (10 Hz)
min_freq: Minimum Frequency (1 Hz)
latency_threshold: Latency Threshold (150 ms)
current_freq: Current Frequency
latency: Network Latency

// Initialization
current_freq = initial_freq

// Begin Adaptive Algorithm
while True do
// Measure Latency
latency = measure_latency()

// Check latency against threshold
if latency ≥ latency_threshold then
// Decrease frequency using multiplicative decrease
current_freq = max(min_freq, current_freq * 0.7)
else
// Increase frequency using additive increase
current_freq = min(max_freq, current_freq + 0.5 Hz)
end if

// Set Data Transmission Frequency
set_data_transmission_frequency(current_freq)

// Wait for the next measurement interval
sleep(subscribe_time_interval)
end while
```

3.2. EDRP

3.2.1. EDRP System

To facilitate efficient data transmission and reception between the robot and the EDRP in the 5G MEC, the MQTT broker was established on the 5G MEC, as shown in Figure 2. This MQTT broker receives the messages published by the custom-built robot and relays them to other software subscribing to those topics. These software components include the localization, local planning, global planning, and perception modules required to realize autonomous driving, and these components subscribe to the required data from the MQTT broker to receive and utilize the data.

Figure 5 illustrates the Kubernetes system of the EDRP. The platform is based on Kubernetes and comprises the autonomous driving modules and the register-api-server. First, the autonomous driving module was designed using the MSA, which addresses the limitations of the system scalability and lack of software flexibility found in the monolithic architecture commonly used in robotics and autonomous driving. It also provides advantages in terms of implementing offloading for multiple robots [34]. The autonomous driving module comprises pods, which are the smallest deployment units in Kubernetes, including the ROS perception pod, ROS global pod, and the ROS local pod, which implement each autonomous driving function, as well as the ROS master pod, which manages these modules.

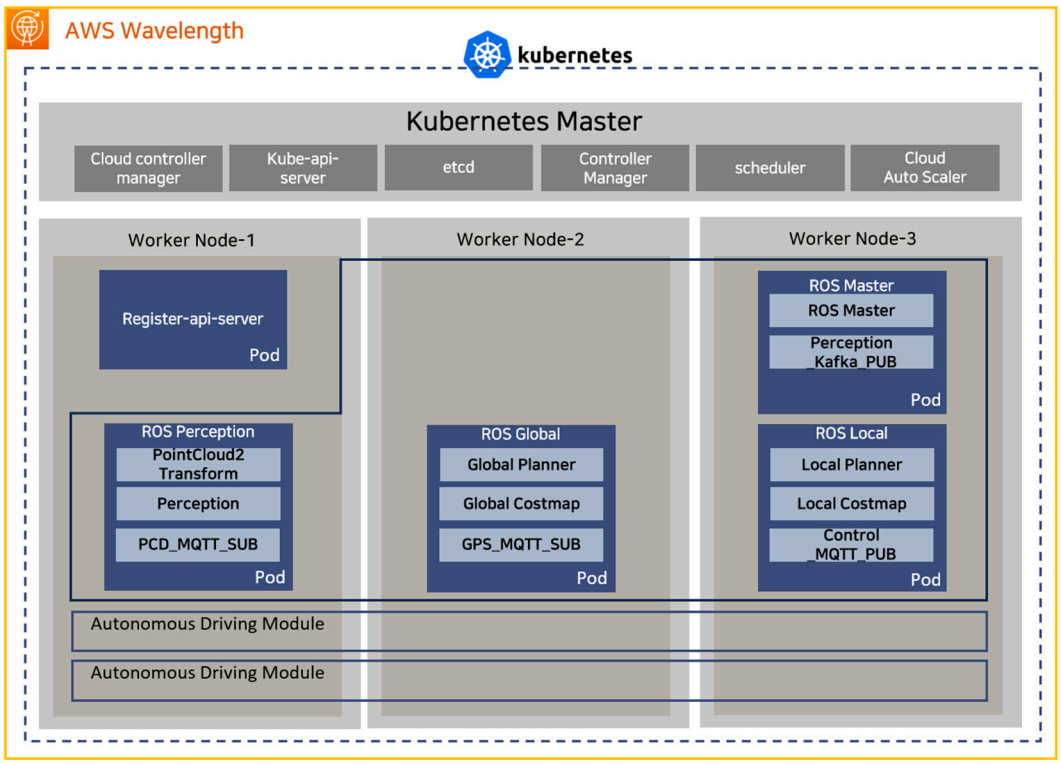

Figure 5. EDRP Kubernetes architecture.

The ROS master pod manages the communication between pods. Here, the perception pod employs nodes for dynamic object detection and environmental awareness based on the point cloud data acquired from the robot. It includes data transformation and object recognition functionality through machine learning. The localization pod is configured to estimate the robot's location based on the GPS data obtained by the robot. Finally, the local planning and global planning pods comprise a planner and a cost map, which help the robot to navigate effectively.

Second, the register-api-server is implemented based on the Flask framework. It deploys the autonomous driving module for each robot offloading process in the Kubernetes environment. Note that this configuration is designed to enable the offloading of tasks for multiple driving robots within a single Kubernetes cluster. In addition, communication between pods in the autonomous driving module is managed through the ROS master deployed in the module. The ROS master pod provides the required ports for the ROS nodes in other pods to find services and communicate with each other. This configuration allows the ROS nodes in each autonomous driving module to discover each other and exchange messages via the ROS master.

Figure 6 shows the logical flow of the proposed system, and Table 4 presents message formats in the EDRP. The explanation sequence is indicated by the numbers.

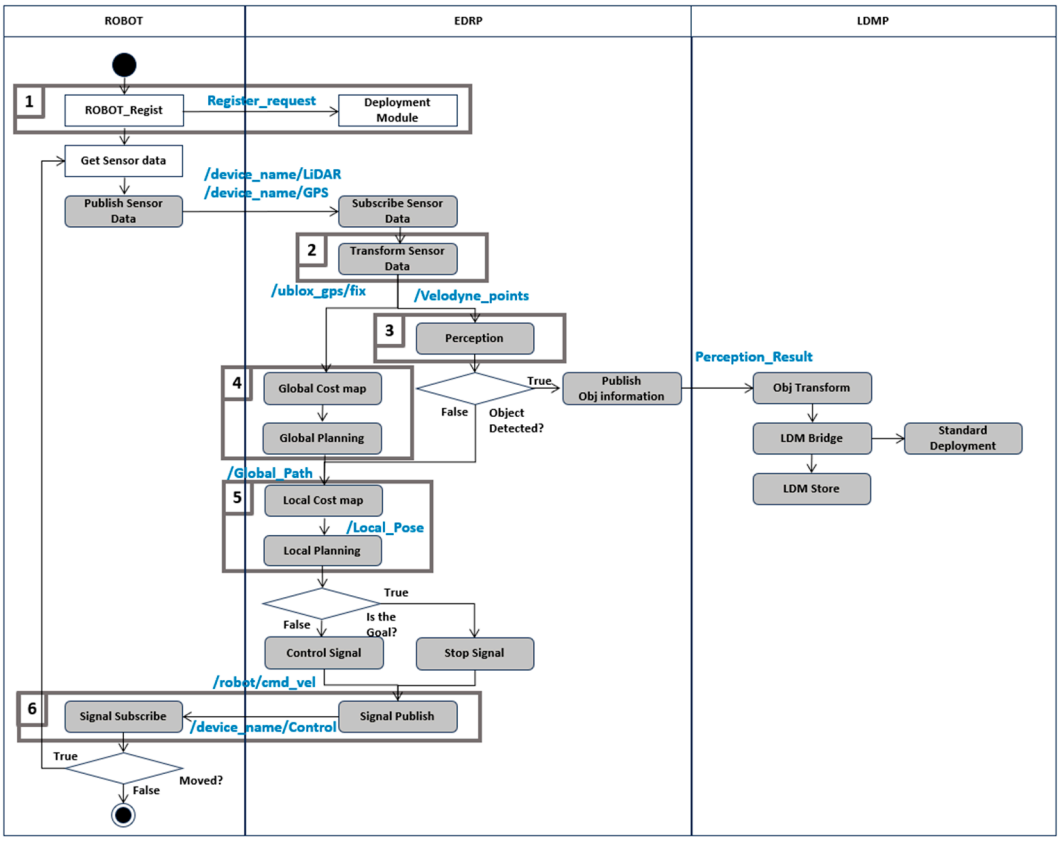

Figure 6. Logical flow of the overall system.

1. To register a robot, the system receives a Register_request postmessage (Table 4) that includes the Device_name, which is used to deploy the autonomous driving module under that name.
2. In the transform sensor data process, the data received via MQTT are converted into ROS topics for use in the autonomous driving module. For localization, the /device_name/GPS data received by the GPS_MQTT_SUB node are converted to /Ublox_gps/fix.
3. For perception, the LiDAR data received as a /device_name/Lidar message are converted to /Velodyne_points. When the perception module identifies dynamic objects, it publishes a Perception_Result message to the LDMP via Kafka. This message includes the PCD data, GPS data, and information about the dynamic object.
4. The global planner then publishes a Global_Path based on the robot's destination.
5. Subsequently, the local planner adjusts the route by integrating the obstacle information acquired from the perception module with the Global_path.

6. Based on the adjusted path, it publishes the /robot/cmd_vel message, which is then converted to /device_name/Control by the Control_mqtt_pub node and sent to the robot.

Table 4. Message formats in EDRP.

Name	Protocol	Format	Description
Register_request	HTTP	POST Request	Registers a robot with the EDRP
Global_Path	ROS Topic	Nav_msgs/Path	Overall path for the robot to follow
Local_Pose	ROS Topic	Geometry_msgs/PoseStamped	The device's current pose with respect to a local frame
/Ublox_gps/fix	ROS Topic	Sensor_msgs/NavSatFix	GPS data (including latitude, longitude, and altitude)
/Velodyne_points	ROS Topic	Sensor_msgs/PointCloud2	LiDAR point cloud data from a device's Velodyne sensor
/robot/cmd_vel	ROS Topic	Geometry_msgs/Twist	Commands for a device's linear and angular velocities
/device_name/Lidar	MQTT	Custom JSON	LiDAR PCD compression data from Velodyne sensor
/device_name/GPS	MQTT	Custom JSON	GPS data from Ublox GPS sensor
/device_name/Control	MQTT	Custom JSON	Control commands to the device
Perception_Result	Kafka	Custom Data Type	Sensor data with object recognition results

3.2.2. EDRP Scalability

Figure 7 shows the scalability of the proposed EDRP. As can be seen, as the number of robots increases, there may be instances where the existing Kubernetes worker nodes cannot deploy additional autonomous driving modules. In such cases, the Kubernetes cluster employs a policy that utilizes the Cluster Autoscaler (CA) to add new nodes. The CA monitors the overall resource usage within the cluster and adds new worker nodes as required, thereby ensuring that the autonomous driving modules in the pending state are deployed to new nodes.

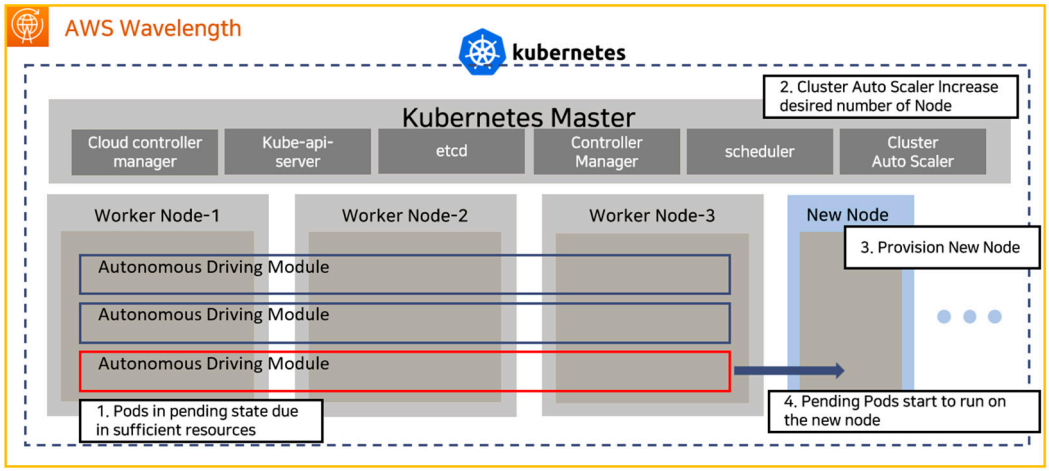

Figure 7. EDRP scalability.

3.3. LDMP

The LDMP is implemented based on the cloud. As shown in the logical flow diagram in Figure 8, the LDMP can be divided into three main components, i.e., the object transform, LDM bridge, and LDM store components.

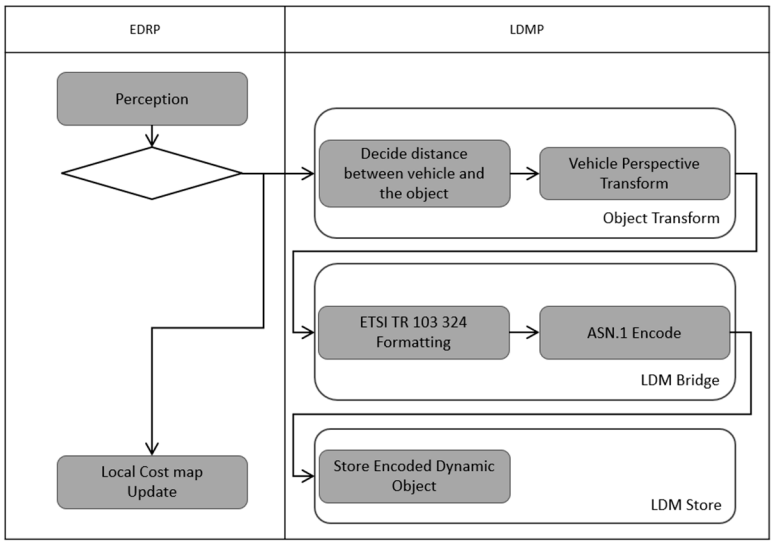

Figure 8. Logical flow of LDMP.

3.3.1. Object Transform

The object transform process involves converting the perception results obtained from the EDRP into a format suitable for storage in the LDM. Here, the data structure for the dynamic objects stored in the LDM follows the CPM format specified in ETSI TS 103 324. This message type is used to share information about objects detected by a local perception sensor, e.g., LiDAR. Please refer to Table 5 for the CPM format. The essential elements for the object distance measurement include the reference position and perceived object types, which include the position of objects relative to the vehicle. To obtain these data from the point cloud, the process involves two steps, i.e., estimating the distance between the sensor and the object, and representing the positional relationship between the object and the vehicle. Note that the distance between the sensor and the object is acquired by filtering out outliers, noise, and the ground plane from the point cloud, applying voxel downsampling to the point cloud, and then calculating the average distance of the obtained points. This distance is then converted into the distance between the vehicle and the object using the positional relationship between the sensor and the center of the vehicle. The transformed distance is then used to represent the object information according to the ETSI TS 103 324 standard for dynamic objects.

Table 5. Collective Perception Message (CPM) format.

Field	Type	Description
CPM Header	Header	ItsPduHeader.
ManagementContainer	referenceTime	Timestamps
	referencePosition	ReferencePosition
	segmatationInfo	MessageSegmentationInfo (Optional)
	messageRateRange	MessageRateRange (Optional)
OriginatingVehicleContainer	orientationAngle	Wgs84Angle
	ptichAngle	CartesianAngle (Optional)
	rollAngle	CartesianAngle (Optional)
	trailerDataSet	TrailerDataSet (Optional)
PerceivedObjectContainer	numberOfPerceivedObjects	CardinalNumber1B
	perceivedObjects	PerceivedObjects Position

3.3.2. LDM Bridge

The data refined through the object transform process undergo a formatting process based on the ETSI TS 103 324 standard to apply the LDM standard [3]. These are converted to the CPM format specified in the ETSI TS 103 324 standard. Then, the dynamic object

detection information is encoded into the ASN.1 format, which ensures compatibility with the proposed platform and with various V2X communication devices produced for the C-V2X environment by different vendors. The encoded message is then published to the /dynamic_object_cpm topic (Table 3) via Kafka.

3.3.3. LDM Store

The LDM store platform was built based on PostgreSQL and a Kafka connector. Here, PostgreSQL constitutes the LDM, serving as a platform to collect dynamic object information. After receiving the /dynamic_object_cpm topic via the Kafka connector, the dynamic object information is transformed and stored in a PostgreSQL database.

3.3.4. LDMP Scalability

As the amount of dynamic object information increases, the capacity of database instances can be adjusted dynamically through the automatic scaling of cloud services. This feature monitors the data workload of the PostgreSQL database and scales up the storage automatically on demand to prevent performance degradation. Thus, this ensures that even when the dynamic object information increases, the system can store and manage data without suffering from any performance issues.

3.4. Security

In this study, TLS encryption, which is a security mechanism of MQTT, was applied to ensure secure data transmission between robots and the edge cloud. Note that TLS encryption is crucial for preventing eavesdropping and data tampering during data transmission. This ensures that communication between robots and the edge cloud is secured, and the confidentiality and integrity of data are maintained. All communications between the MQTT broker and clients are encrypted using TLS, thereby minimizing the possibility of data interception or tampering. A trustworthy communication channel can be established using certificates for mutual authentication between clients and brokers, which is particularly important in autonomous driving robot systems that handle sensitive data.

TLS encryption, which is a security mechanism of Kafka, was applied to secure the transmission of dynamic object data via the LDMP. By applying TLS encryption, communications between brokers as well as between clients and brokers are protected, which is essential to prevent eavesdropping and data tampering during data transmission. Kafka's TLS encryption maintains the confidentiality and integrity of data transmission, ensuring that sensitive dynamic object data are securely protected.

The communication between robots and the edge cloud as well as the transmission of dynamic object data via the LDMP are secured, thereby enhancing the overall security of the system. This ensures that the proposed system can respond effectively to potential security threats, providing reliability and safety. Applying such security mechanisms to maintain the confidentiality, integrity, and availability of data is essential because it enhances the overall reliability of the system considerably.

4. Experiment

This section discusses an experimental evaluation of the proposed platform. To verify the proposed platform, we implemented the device, EDRP, and LDMP. Based on these independent implementations, we conducted performance tests using a use case to evaluate the applicability and interconnectivity of the proposed platform.

4.1. Platform Implementation

The proposed platform is primarily composed of the robot, EDRP, and LDMP, as shown in the Figure 3. Here, the robot offloads its driving functions. The EDRP is a Kubernetes-based autonomous driving offloading system, and the LDMP distributes and stores information about the dynamic objects. With platform implementation, performance

tests were conducted to assess the overall functionality and integration of these components of the proposed platform.

4.1.1. Device

As shown in Figure 9, we constructed a robot to perform autonomous driving offloading. The hardware components of the robot are detailed in Table 6. For position information, we used the ZED-F9P model from Ublox, which is a high-performance GNSS module that provides centimeter-level precise positioning. This module is suitable for autonomous vehicles and robots requiring precise location data due to its real-time kinematic capabilities. To collect data on the surrounding environment, we utilized a VLP-16 LiDAR sensor from Velodyne. This sensor measures the environment using laser beams and calculates distances to generate 3D point clouds, and it is widely used in autonomous vehicles, robots, drones, and various industrial fields. For 5G internet connectivity, we connected a Galaxy Fold 2 mobile device equipped with a Qualcomm Snapdragon X55 5G modem via USB tethering. This configuration allowed us to access the AWS Wavelength directly over the 5G network. The robot was also equipped with a laptop computer running Ubuntu 20.04 with the ROS Noetic middleware. Note that the laptop was configured to acquire sensor data and control the robot through RS-485 communication to the motor driver.

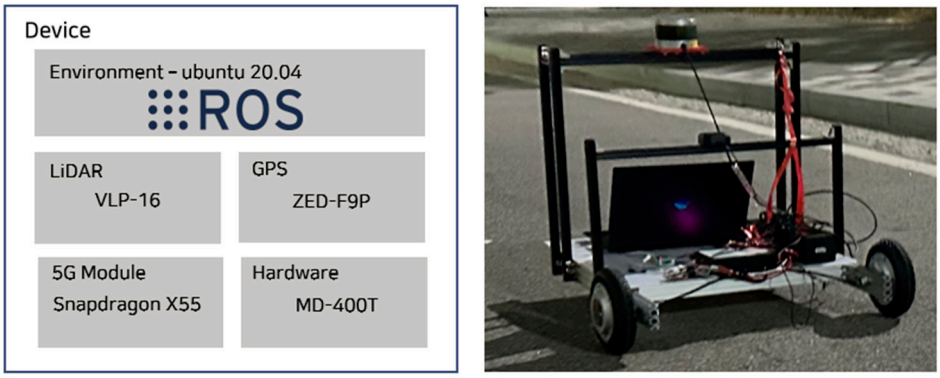

Figure 9. Driving robot system.

Table 6. Robot system information.

GPS	LiDAR	Mobile Communication	Hardware Base	Device Laptop
SPARKFUN ZED-F9P RTK	VLP-16	Qualcomm Snapdragon X55 5G Modem	MD400T MDH-220	I7-10750H RTX2070

4.1.2. EDRP

Figure 10 shows the environment implemented for the EDRP and implemented on the AWS Wavelength's EC2, which is 5G MEC. We established both a Kubernetes cluster and an EMQX MQTT broker on this platform. Note that each function was containerized and deployed on Kubernetes.

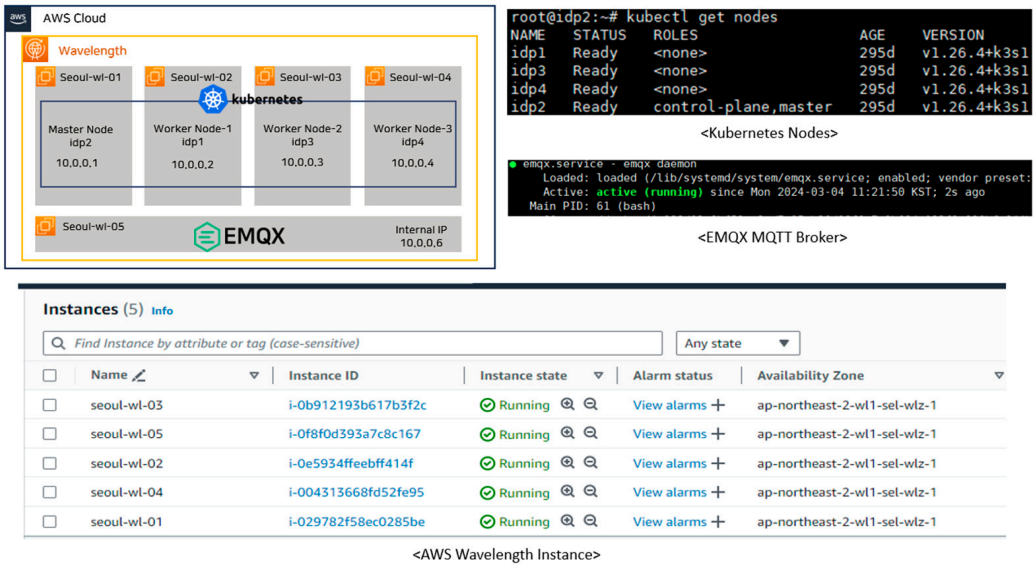

Figure 10. EDRP environment.

4.1.3. LDMP

Figure 11 presents the environment developed for the LDMP. The LDMP performs object transform, LDM bridge, and LDM store processes and was built using a Kafka cluster and a PostgreSQL database on AWS EC2. When a dynamic object is detected by the EDRP, the data are sent to the LDMP via Kafka. The object transform component preprocesses the data for storage and distribution. Then, the data undergo ASN.1 encoding through the LDM bridge and were distributed to various C-ITS components via Kafka. The LDM store receives the dynamic object data through the Kafka PostgreSQL connector and stores the data in the PostgreSQL database. This configuration allows for efficient processing, encoding, and storage of dynamic object information, thereby facilitating real-time data sharing and integration in the C-ITS environment.

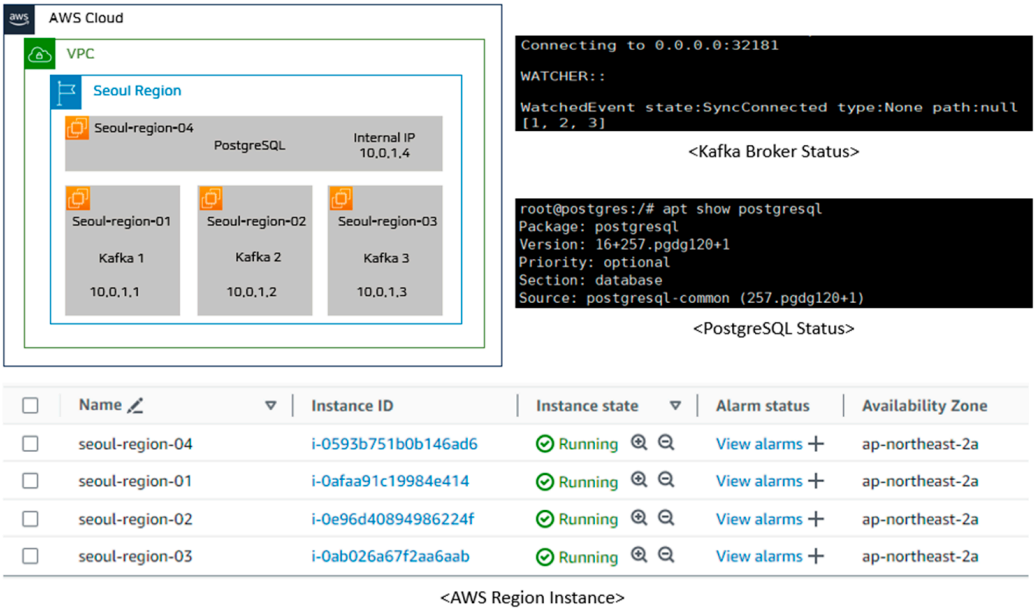

Figure 11. LDMP environment.

4.2. Performance Evaluation

We designed the experiments to verify the integration and interoperability of the EDRP and LDMP. The experimental setup focused on offloading the driving robot platform and storing data in the LDMP by transmitting and receiving actual LiDAR and GPS data, as well as the robot control messages.

The experiment was conducted by generating all route points based on Lanelet2, using the red arrow in Figure 12. The experimental scenario for the qualitative evaluation is summarized as follows.

Figure 12. Driving route.

(1) Confirm that the data acquired from the robot platform are transmitted to the EDRP successfully.
(2) Verify that objects are recognized correctly by the perception node.
(3) Ensure that the global planner generates the shortest path to the target point.
(4) Check that the local planner generates the control signals for the vehicle based on the pure pursuit algorithm.
(5) Confirm that the robot platform follows the designated path based on the control signals from the local planner.
(6) Verify that dynamic objects detected by the perception node in the yellow box area (Figure 12) are stored in the LDMP.

4.3. Qualitative Evaluation

4.3.1. Deployment of Autonomous Driving Module

Figure 13 shows the results of sending a register request with Device_name to register the API server. The middle image in Figure 13 illustrates the process of sending the register request. Figure 13 depicts the final image, where the registered API server has used Device_name to generate the autonomous driving module for driving offloading. This confirms that the Offloading Application is successfully deployed on the EDRP through the robot's register request post.

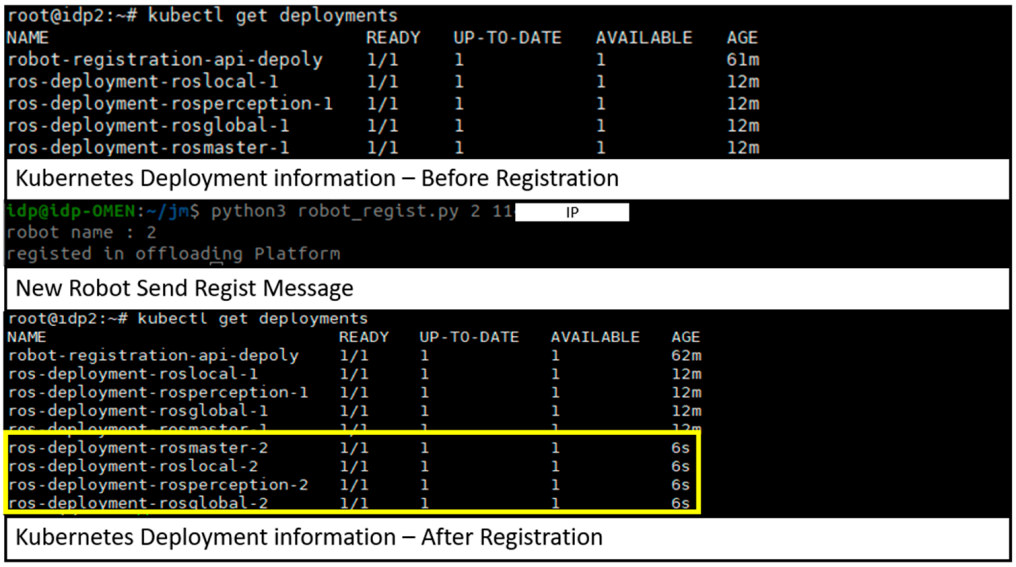

Figure 13. Example of the Kubernetes environment before and after robot registration.

4.3.2. Path Creation, Vehicle Control Signal, and Robot Driving

Figure 14 shows the test robot and its actual motion, indicated using yellow box on the middle and right figures. This driving test confirmed that the robot's sensor data were transmitted to the EDRP successfully. The EDRP generated the entire route based on Lanelet2 and created appropriate control signals for the vehicle through the local planner. By receiving these control signals and confirming that the robot physically drove, we verified that the proposed EDRP operates as expected.

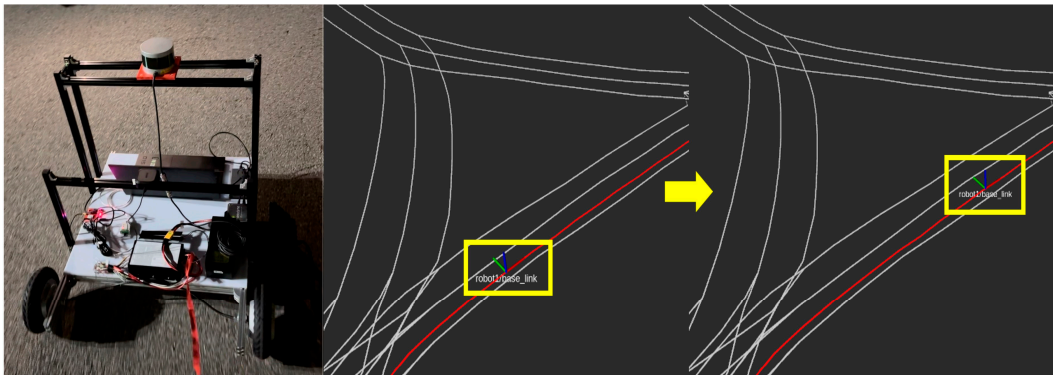

Figure 14. Photograph of the test robot and its sample movement route.

Figure 15 shows the process used by the EDRP to generate a path while avoiding obstacles along the route. The EDRP employs the A* algorithm to identify the optimal path around obstacles. In the figure, red line represents the path traveled by the robot, and green dot denotes the obstacle positions. The robot uses the A* algorithm to avoid obstacles in real time and generate an efficient path to the target destination.

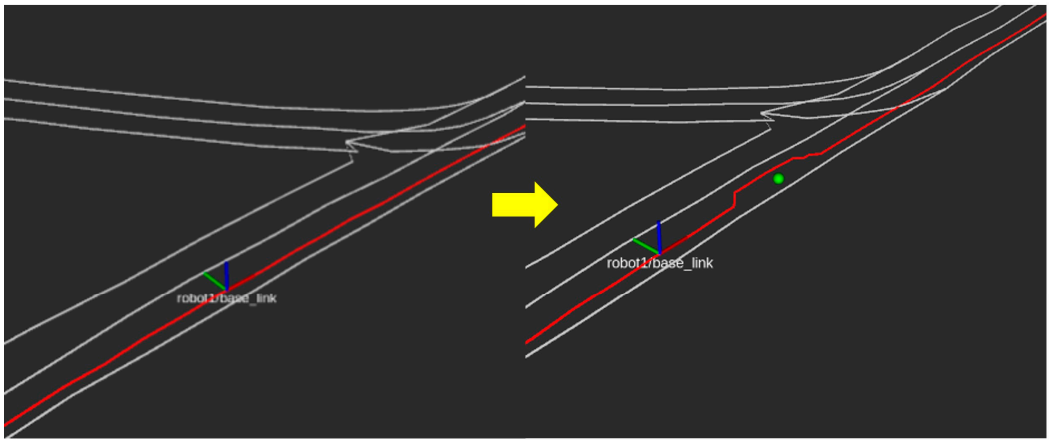

Figure 15. Sample obstacle avoidance.

4.3.3. Verification of EDRP Robot Operational Scalability

Figure 16 demonstrates the operation of the robots. Here, robot1 was the actual robot driving on the road, while the other robots were virtual robots operated on the EDRP. As can be seen, the proposed EDRP can manage and operate five robots simultaneously. The experimental results verified that the proposed EDRP can handle multiple robots, thereby providing a scalable solution for robot driving systems. When operating a total of three robots, the Kubernetes environment's GPU, established for this experiment, showed an average usage rate of approximately 50%. When operating five robots, the average GPU usage rate increased to around 70%. This demonstrates that the EDRP built for this experiment can successfully manage and operate up to five robots.

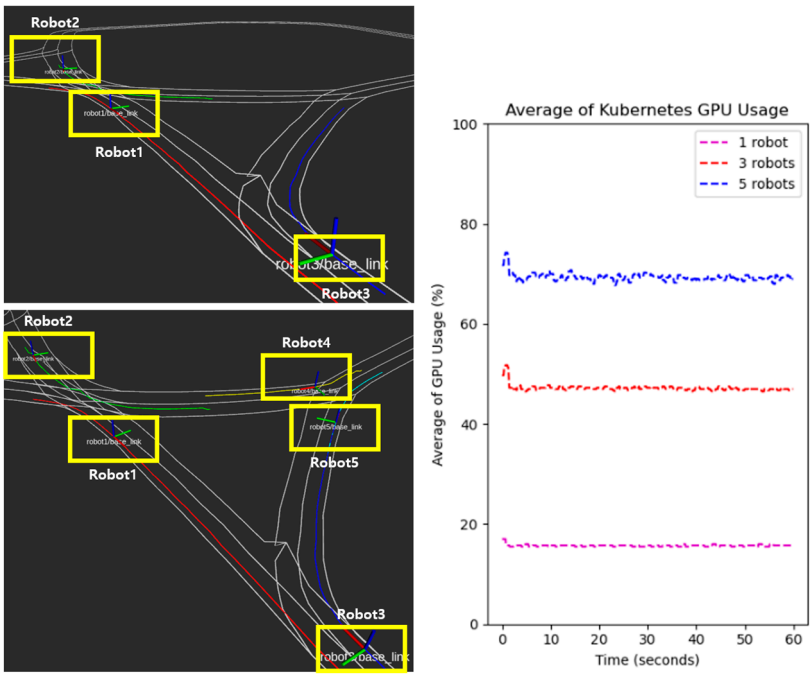

Figure 16. EDRP multirobot operation and GPU usage graph.

4.3.4. Object Perception and LDMP Storage

Figure 17 shows the result of detecting a dynamic object and a pedestrian on the EDRP using the PV-RCNN model from openPCDet for point cloud object detection [35]. Data for the detected pedestrian are processed using the LDM bridge. As seen in the middle part of Figure 17, information regarding the dynamic object was converted according to the ETSI

TS 103 324 standard. The bottom image in Figure 16 demonstrates the result of publishing the message, which was encoded in the ASN.1 format based on this standard, via Kafka.

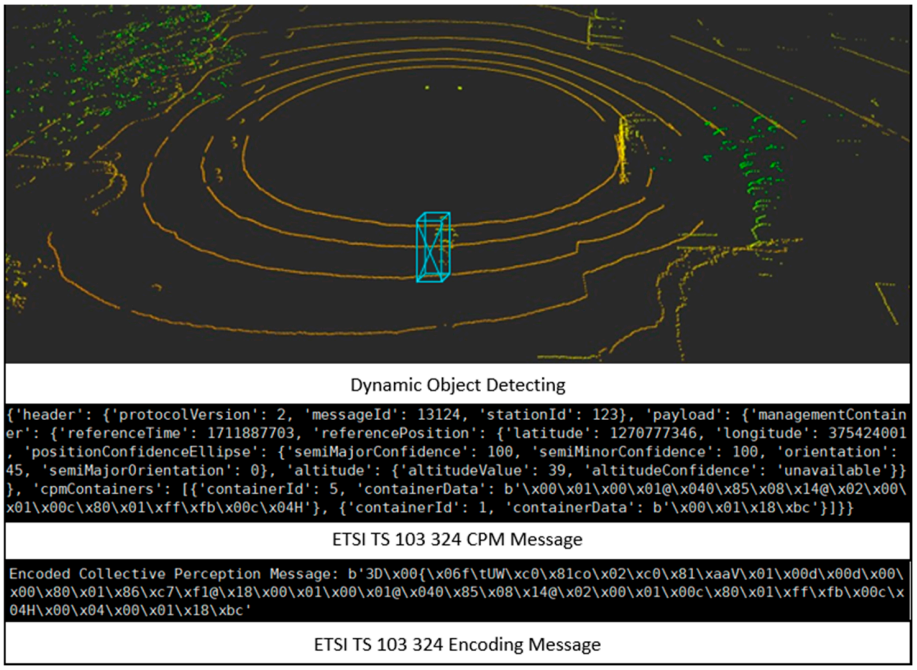

Figure 17. Object detection and a CPM example.

Figure 18 displays a sample map with a dynamic object and the PostgreSQL database. In the PostgreSQL database, the encoded messages received via the Kafka connector are decoded and stored in a table, as shown in Figure 18. The stored data were then visualized on the map, where the red point within the red bounding box indicates the successfully stored and visualized object. This demonstrates that the Kafka messages published by the LDMP can be used in various applications in C-ITS components.

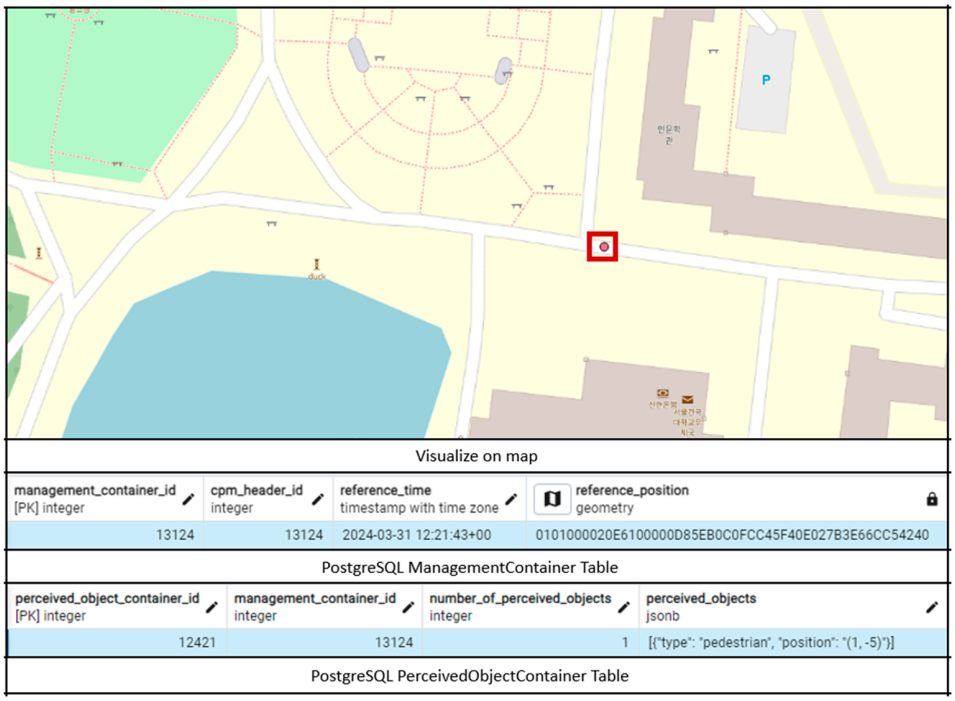

Figure 18. Example of a map with dynamic object and PostgreSQL table.

4.4. Quantitative Evaluation

4.4.1. Energy Efficiency

The left graph in Figure 19 compares the memory usage over time. The offloading measurement used lower memory than onboard processing. The onboard measurement used more than 18% of the memory, while the offloading measurement utilized only approximately 12%. The middle graph compares the GPU usage over time. As shown, the offloading measurement exhibited lower GPU usage than onboard processing. The onboard measurement used greater than 60% of the GPU, while the offloading measurement utilized less than 10%. Finally, the right graph in Figure 19 compares the power usage over time. The offloading measurement used lower power than onboard processing. Here, the onboard measurement used approximately 70–80 W of power. Meanwhile, the offloading measurement used only approximately 20 W.

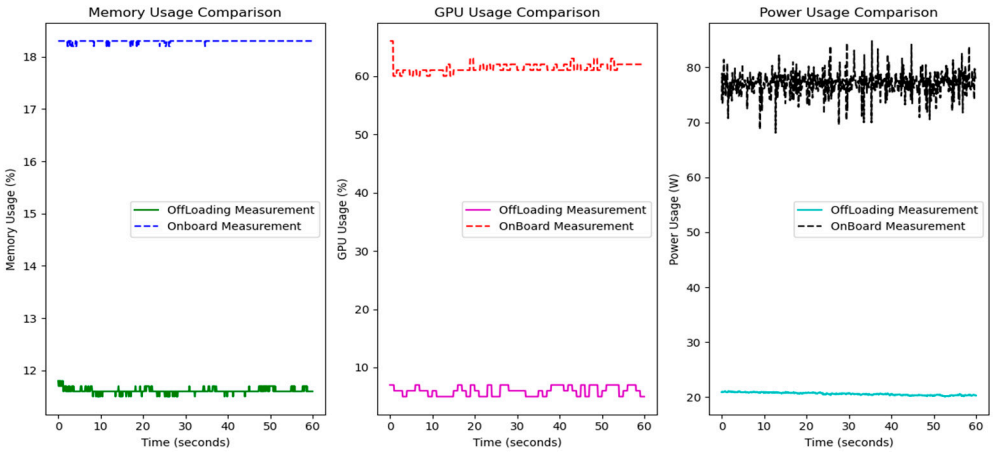

Figure 19. Comparison of resource usage graph.

These results revealed that the proposed system can provide the same functionality as onboard processing but with lower memory, GPU, and power usages, which reduce battery consumption and increase the operational time for actual driving robots.

4.4.2. EDRP

To perform a quantitative evaluation, a round trip time (RTT) was measured from the time the robot's sensor data were sent, processed through the autonomous driving computation, and the control signal was received by the robot. In Figure 20, we present the RTT for the proposed EDRP, which averaged at approximately 145 ms.

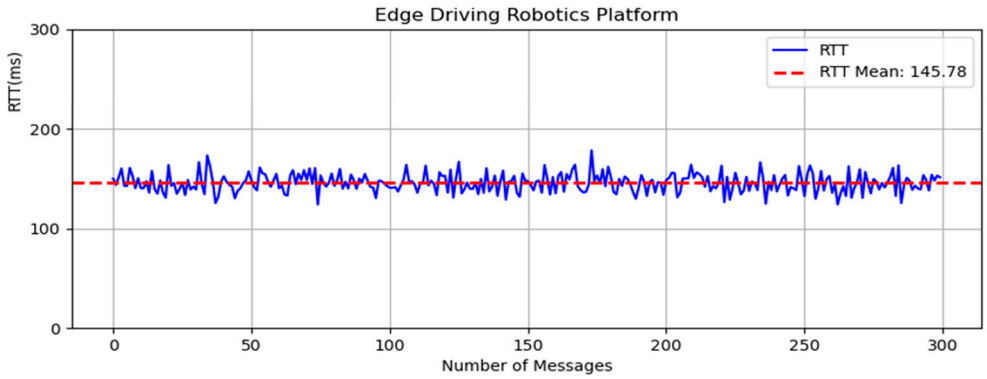

Figure 20. Edge-driving robotics round trip time (RTT) graph.

To evaluate the effectiveness of applying Kubernetes to the proposed EDRP, an offloading environment without Kubernetes was implemented in a 5G MEC environment, and the RTT was measured. Figure 21 shows the RTT results, which averaged at approximately 142 ms in this case. Compared with the proposed EDRP, this average RTT differed by approximately 3–4 ms. This means that offloading the software of multiple robots as Kubernetes pods in a multirobot environment is practically feasible. The 145 ms latency demonstrates a faster response time than the maximum human reaction time of 150 ms when driving [36].

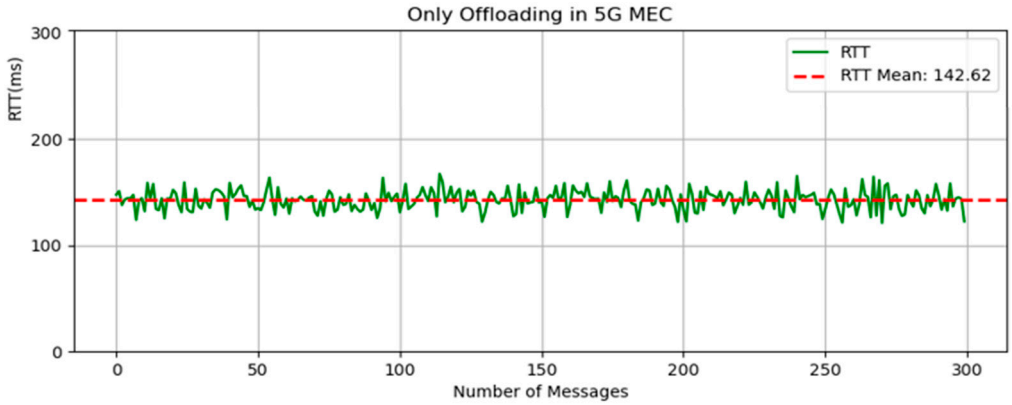

Figure 21. Offloading only in 5G MEC RTT graph.

In addition, according to Zhang et al., for remote vehicle teleoperation, a total latency of less than 170 ms has minimal impact on teleoperation performance and does not cause control issues when driving remotely. Thus, we consider that the proposed EDRP is a viable solution [37]. The latency in a Kubernetes-based cluster environment for offloading and operating multiple robots is within an acceptable range. Thus, the proposed platform ensures applicability in multirobot environments and can be operated reliably in real-world driving scenarios.

4.4.3. LDMP

After recognizing dynamic objects, the latency of publishing the CPM via Kafka according to ETSI TS 103 324 was measured, as presented in Figure 22. Here, the transmission latency of the encoded CPM containing dynamic object information was approximately 42 ms on average. This result indicates that the C-ITS components and the LDM store, which stores the dynamic object information, can receive the information with low latency. Thus, the proposed EDRP and LDMP demonstrate sufficient performance for offloading driving robots and sharing dynamic object information.

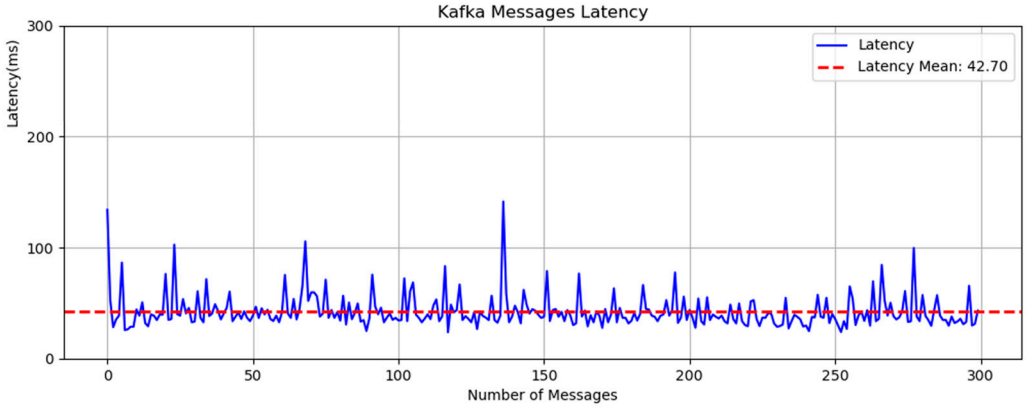

Figure 22. Kafka messages latency graph.

5. Conclusions

This study has proposed a system that combines the EDRP and LDMP to address the issues of short operational time of autonomous robots due to large sensor data processing, high setup costs, and the lack of dynamic object information sharing with surrounding autonomous robots and other mobility systems. In the proposed system, the data acquired by the robots are received by the EDRP and deployed within the 5G MEC. Then, the control signals generated based on this data are sent back to the robots.

The overall RTT of this process was found to be 145 ms, which confirms the feasibility of the robot offloading process. In addition, the autonomous driving module, which was designed with the MSA, can be operated and deployed for each robot via Kubernetes when offloading two or more autonomous robots.

In addition, the LDMP for autonomous robots collects and shares dynamic object data based on C-ITS standards from each autonomous robot's offloading environment, thereby allowing for a wider range of environmental awareness compared to operating a single driving robot. Sharing dynamic object information in real-time based on the ETSI TS 103 324 standard provides both integration and compatibility with C-ITS components.

Compared with the traditional onboard processing of robots, the proposed EDRP showed 30% reduction in memory usage, 80% reduction in GPU usage, and 70% reduction in power consumption for computation. This can lead to reduced robot construction costs through offloading computations and increased operational time owing to decreased computational processes. Additionally, high-performance AI can be easily applied by deploying it on the EDRP without requiring computation resource upgrades. The LDMP can also enhance the utility of driving robots in various practical applications, e.g., disaster response, security, and environmental monitoring.

The findings of this study and expected improvements from the future work can enable the establishment of a safe and efficient offloading, driving robot environment by offloading processes and sharing dynamic object information.

The limitations of this study and future work are as follows. First, this study conducted tests based on simple autonomous driving applications due to the limitations of the proposed EDRP. The feasibility of offloading and driving based on simple applications was confirmed; however, more advanced autonomous driving scenarios must be offloaded and tested prior to exploring commercialization.

Second, the LDMP transmits information about dynamic objects without considering the receiver's location and the environment. Therefore, to realize efficient dynamic object sharing, further research on filtering and selecting information based on the receiver's location and environment is required.

Third, this study was conducted based on a stable network environment. As the demand for robots increases, appropriately allocating and optimizing the resources of each network node become essential. Therefore, further research is required to apply various technologies, e.g., network slicing, to prepare for unstable network environments. Additionally, network management and optimization strategies must be considered in future studies.

6. Future Work

6.1. System Expansion and Optimization

6.1.1. 6G Network

The application of 6G networks will enable ultralow latency communication, thereby dramatically reducing data transmission times between robots and edge servers. In addition, 6G networks are based on satellite communication; thus, future research will focus on leveraging this capability to ensure that EDRP can operate in remote or underserved regions. This could be particularly beneficial for applications in environmental monitoring, disaster response, and security operations. By leveraging 6G's capabilities, the proposed system might maintain robust and continuous communication, potentially ensuring reliable data transmission and operational efficiency regardless of location.

6.1.2. 5G Specialized Network

The 5G specialized network environment is suitable for operating robots in remote areas where commercial networks cannot be used or in military and industrial environments where security is paramount. The 5G specialized network provides an independent communication environment, ensuring stable connections and a secure environment for operating the proposed system. Future work will explore the integration of 5G specialized networks to enhance the stability and security of the proposed system.

6.2. Applicability

By applying the proposed system, search and rescue robots can process data in real-time and respond to various situations in disaster sites quickly. This contributes to saving lives and recovering from damages in disaster events. The extended operational time of robots allows for the exploration of wider areas and performance of rescue operations, thereby enhancing work efficiency through real-time data sharing and collaboration.

In addition, the proposed system can be applied to patrol and guide robots in public spaces. Complex AI tasks that cannot be processed by robots, such as those required by large language models (LLMs), can be performed easily through edge offloading without upgrading the robot's computing resources. This is expected to enable the provision of various additional functions, e.g., user response and patrol using LLMs.

Author Contributions: Conceptualization, J.M. and C.M.; methodology, J.M. and D.H.; software, J.M. and J.K. and D.H.; validation, J.M. and S.K.; investigation, J.M. and D.H. and J.K.; resources, C.M.; data curation, J.M. and D.H.; writing—original draft preparation, J.M.; writing—review and editing, J.M., J.K., C.M., S.W. and H.C.; visualization, J.M.; supervision, C.M.; project administration, J.M.; funding acquisition, C.M. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0020536, HRD Program for Industrial Innovation).

Data Availability Statement: Data is available from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADTFC	adaptive data transmission frequency control
CA	cluster autoscaler
C-ITS	cooperative-intelligent transport systems
CPM	collective perception message
EDRP	edge-driving robotics platform
ITS	intelligent transport system
LDM	local dynamic map
LDMP	local dynamic map platform
MEC	mobile edge computing
MQTT	message queuing telemetry transport
MSA	microservice architecture
ROS	robot operation system
RTT	round trip time

References

1. Corcoran, P.; Datta, S.K. Mobile-Edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network. *IEEE Consum. Electron. Mag.* **2016**, *5*, 73–74. [CrossRef]
2. Agriesti, S.; Gandini, P.; Marchionni, G.; Paglino, V.; Ponti, M.; Studer, L. Evaluation approach for a combined implementation of Day 1 C-ITS and truck platooning. In Proceedings of the IEEE 87th Vehicular Technology Conference (VTC Spring), Porto, Portugal, 3–6 June 2018; pp. 1–6.
3. *ETSI TS 103 324*; Intelligent Transport System (ITS); Vehicular Communications; Basic Set of Applications; Specification of the Collective Perception Service. ETSI: Sophia Antipolis, France, 2020.

4. Spinelli, F.; Mancuso, V. Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility. *IEEE Commun. Surv. Tutor.* **2020**, *23*, 596–630. [CrossRef]
5. Kerr, J.; Nickels, K. Robot Operating Systems: Bridging the gap between Human and Robot. In Proceedings of the 2012 44th Southeastern Symposium on System Theory (SSST), Jacksonville, FL, USA, 11–13 March 2012; pp. 99–104.
6. Hellmund, A.M.; Wirges, S.; Taş, Ö.Ş.; Bandera, C.; Salscheider, N.O. Robot Operating System: A Modular Software Framework for Automated Driving. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 1564–1570.
7. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source robot operating system. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
8. Poniszewska-Marańda, A.; Czechowska, E. Kubernetes Cluster for automating software production environment. *Sensors* **2021**, *21*, 1910. [CrossRef] [PubMed]
9. Jeffery, A.; Howard, H.; Mortier, R. Rearchitecting Kubernetes for the Edge. In Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking (EdgeSys '21), Edinburgh, UK, 26 April 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 7–12. [CrossRef]
10. Abdollahi Vayghan, L.; Saied, M.A.; Toeroe, M.; Khendek, F. Deploying microservice based applications with Kubernetes: Experiments and lessons learned. In Proceedings of the IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; pp. 970–973.
11. Thönes, J. Microservices. *IEEE Softw.* **2015**, *32*, 116. [CrossRef]
12. De Lauretis, L. From Monolithic Architecture to Microservices Architecture. In Proceedings of the 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, 27–30 October 2019; pp. 93–96.
13. Yassein, M.B.; Shatnawi, M.Q.; Aljwarneh, S.; Al-Hatmi, R. Internet of Things: Survey and Open Issues of MQTT Protocol. In Proceedings of the 2017 International Conference on Engineering & MIS (ICEMIS), Monastir, Tunisia, 8–10 May 2017; pp. 1–6.
14. Thandavarayan, G.; Sepulcre, M.; Gozalvez, J. Analysis of Message Generation Rules for Collective Perception in Connected and Automated Driving. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 134–139.
15. Shimada, H.; Yamaguchi, A.; Takada, H.; Sato, K. Implementation and evaluation of local dynamic map in safety driving systems. *J. Transp. Technol.* **2015**, *5*, 102–112. [CrossRef]
16. Zhai, Y.; Ding, B.; Zhang, P.; Luo, J.; Wu, Q.; Shi, P.; Wang, H. Cooperative offloading for multiple robot applications. In Proceedings of the IEEE International Conference on Joint Cloud Computing, Oxford, UK, 3–6 August 2020; pp. 63–70.
17. Wang, W.; Qu, R.; Liao, H.; Wang, Z.; Zhou, Z.; Wang, Z.; Mumtaz, S.; Guizani, M. 5G MEC-based intelligent computation offloading in power robotic inspection. *IEEE Wirel. Commun.* **2023**, *30*, 66–74. [CrossRef]
18. Liu, B.; Wang, L.; Liu, M. RoboEC2: A novel cloud robotic system with dynamic network offloading assisted by amazon EC2. *IEEE Trans. Automat. Sci. Eng.* 2023; 1–15, *Early Access*.
19. Zhang, D.; Zhang, Z.; Zhang, J.; Zhang, T.; Zhang, L.; Chen, H. UAV-assisted task offloading system using dung beetle optimization algorithm & deep reinforcement learning. *Ad Hoc Netw.* **2024**, *156*, 103434.
20. Sossalla, P.; Rischke, J.; Nguyen, G.T.; Fitzek, F.H.P. Offloading robot control with 5G. In Proceedings of the 19th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 461–464.
21. De Souza, A.M.; Oliveira, H.F.; Zhao, Z.; Braun, T.; Loureiro, A.A.F.; Villas, L.A. Enhancing sensing and decision-making of automated driving systems with multi-access edge computing and machine learning. *IEEE Intell. Transp. Syst. Mag.* **2022**, *14*, 44–56. [CrossRef]
22. Ho, T.M.; Nguyen, K.K.; Cheriet, M. Optimized Task Offloading in UAV-Assisted Cloud Robotics. In Proceedings of the ICC 2023—IEEE International Conference on Communications, Rome, Italy, 28 May–1 June 2023; pp. 4773–4779.
23. Wei, P.; Feng, W.; Wang, Y.; Chen, Y.; Ge, N.; Wang, C.X. Joint mobility control and MEC offloading for hybrid satellite-terrestrial-network-enabled robots. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 8483–8497. [CrossRef]
24. Tang, J.; Yu, R.; Liu, S.; Gaudiot, J.L. A container based edge offloading framework for autonomous driving. *IEEE Access* **2020**, *8*, 33713–33726. [CrossRef]
25. Kong, P.Y. Computation and sensor offloading for cloud-based infrastructure-assisted autonomous vehicles. *IEEE Syst. J.* **2020**, *14*, 3360–3370. [CrossRef]
26. Srinivasa, R.; Naidu, N.K.S.; Maheshwari, S.; Bharathi, C.; Hemanth Kumar, A.R. Minimizing Latency for 5G Multimedia and V2X Applications using Mobile Edge Computing. In Proceedings of the 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), Jaipur, India, 28–29 September 2019; pp. 213–217.
27. Yu, Y.; Lee, S. Remote driving control with real-time video streaming over wireless networks: Design and evaluation. *IEEE Access* **2022**, *10*, 64920–64932. [CrossRef]
28. Coronado, E.; Cebrian-Marquez, G.; Riggio, R. Enabling Computation Offloading for Autonomous and Assisted Driving in 5G Networks. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
29. Ayoub, F.; Villing, R. Evaluating Distributed Computation Offloading Scalability for Multiple Robots. In Proceedings of the 2023 Eighth International Conference on Fog and Mobile Edge Computing (FMEC), Tartu, Estonia, 18–20 September 2023; pp. 72–79.

30. Zhu, A.; Zeng, Z.; Guo, S.; Lu, H.; Ma, M.; Zhou, Z. Game-theoretic robotic offloading via multi-agent learning for agricultural applications in heterogeneous networks. *Comput. Electron. Agric.* **2023**, *211*, 108017. [CrossRef]
31. Shin, S.; Kim, J.; Moon, C. Road dynamic object mapping system based on edge-fog-cloud computing. *Electronics* **2021**, *10*, 2825. [CrossRef]
32. Yoo, A.; Shin, S.; Lee, J.; Moon, C. Implementation of a sensor big data processing system for autonomous vehicles in the C-ITS environment. *Appl. Sci.* **2020**, *10*, 7858. [CrossRef]
33. Cho, K.; Cho, D. Autonomous driving assistance with dynamic objects using traffic surveillance cameras. *Appl. Sci.* **2022**, *12*, 6247. [CrossRef]
34. Blinowski, G.; Ojdowska, A.; Przybyłek, A. Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access* **2022**, *10*, 20357–20374. [CrossRef]
35. OD TEAM. Openpcdet: An Open-Source Toolbox for 3d Object Detection from Point Clouds. 2020. Available online: https://github.com/open-mmlab/OpenPCDet (accessed on 12 January 2023).
36. Jayaweera, N.; Rajatheva, N.; Latva-aho, M. Autonomous driving without a Burden: View from Outside with Elevated LiDAR. In Proceedings of the IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Kuala Lumpur, Malaysia, 28 April–1 May 2019; pp. 1–7.
37. Zhang, T. Toward automated vehicle teleoperation: Vision, opportunities, and challenges. *IEEE Internet Things J.* **2020**, *7*, 11347–11354. [CrossRef]