Load Distribution and Service Time Analysis of a Vehicular Edge Computing System

Thone Thone Win Maw
Department of Computer Sci. and Eng.
Chungnam National University
Daejeon, South Korea
ttwinmaw@o.cnu.ac.kr

Hoon Choi
Department of Artificial Intelligence
Chungnam National University
Daejeon, South Korea
hc@cnu.ac.kr

***Abstract*—Due to the evolution of vehicles, most of them have some amount of computational capacity. Vehicular Edge Computing (VEC) has been introduced in recent years due to its augmentation of computational capacity. With the advance of VEC, vehicles are used to reduce latency and improve the quality of service (QoS) of network-based information services. In this paper, vehicles are used as edge servers and are grouped as clusters depending on the service groups that contain popular application services. Whenever a mobile device requests a service to the base station, the edge gateway in the base station sends the request to the cluster that has the requested service. This paper provides the load distribution mechanism within the cluster using the Deep Q Network. Moreover, the total processing time on the edge servers is compared with that of the cloud servers. It was shown that processing in the edge server is better than the cloud server when the length of the request message is short, or the wireless transmission speed is fast.**

***Index Terms*—vehicular edge computing, Deep Q Network, edge server, cloud server**

I. Introduction

Vehicles and the mobile communication infrastructures such as Roadside Units (RSUs) and Base Stations can contribute computational resources. They form the Vehicular Edge Computing System [1]∼ [3] which can process tasks from autonomous driving systems and various applications from pedestrians' mobile devices.

Due to resource constraints, mobile devices need the help of a cloud server when running resource-intensive applications. Therefore, as the number of service requests to the cloud server increases, the network traffic also exacerbates. These situations can get worse when billions of devices are deployed worldwide [4] [5]. To satisfy the required Quality of Service (QoS) is another challenge for cloud server.

To address these issues, Vehicular Edge Computing (VEC) is utilized as a middle layer between mobile devices and cloud server. Hou et al. [6] were the first to introduce the concept of Vehicular Fog Computing (VFC) as an architecture that can be used to enable multiple end-user or edge devices to collaborate to carry out computation and communication tasks. VEC furthers the benefits yielded by cloud computing services to the edge of the network [7]∼ [9]. The increasing number of vehicles on the road also motivates to create an efficient collaboration among vehicles and the concept of VEC where vehicles are utilized as edge servers and can serve the role of service providers [10] [11].

In VEC, the processing of the task takes place in the edge server (vehicle) which is in proximity to the mobile devices on behalf of the cloud server. As the processing of the task is provided close to the mobile device, better QoS is offered by the edge server than the cloud server.

In [12], RSUs serve as edge servers and whenever the edge servers receive any request from end devices (vehicles), the available resources are checked. If these resources in the requested edge servers are not enough resources to process the task, then the offloading is requested to SDN controller (edge gateway). The authors used Deep Q network-based reinforcement learning to select the resources-rich edge server in VANET and proposed that the estimation of vehicle's next location can help in the optimal offloading of processing requests in current edge servers [12]. In [13], a cluster-enabled capacity-based load-balancing approach is described to operate performance-aware vehicular edge distributed computing for efficiently processing IoT jobs, and a clustering approach that considers the position, speed, and direction of vehicles (edge servers) to form their clusters that act as the pool of computing resources.

In this paper, vehicles serve as edge servers and the base station works as an edge gateway, which dispatches the task requests from mobile devices to an appropriate edge server. Moreover, unlike [13], the clustering is based on the services that are already installed in the edge servers. When a service task is requested from the mobile device to the edge gateway, the edge gateway finds the cluster for the requested service and chooses the two optimal edge servers in that cluster using Deep Q Network. If no cluster has the service of the task or both of the two optimal edge servers in the cluster are overloaded, the edge gateway will send the task directly to the cloud server. In [12], Deep Q Network is used to select the optimal edge server to offload the task from the current edge server, which is too overloaded to process the tasks requested by the vehicles. In our proposed design, Deep Q Network reinforcement learning is used to distribute the loads among the edge servers in the clusters and to select the optimal edge server in the cluster when the mobile devices request the tasks. In [12], the state vectors used in Deep Q Network are slightly different from

our proposed design, and it is explained in detail in Section 2.

Our work aims to utilize the aggregate mobility behavior of vehicles to select reliable vehicle nodes and avoid service failure and give better performance than cloud servers.

The rest of the paper is organized as follows: Section 2 presents the problem formulation of our proposed design. Section 3 formalizes the total processing time using Edge Computing and Cloud server. In Section 4, we show the numerical results to compare the total processing time between Edge Computing and Cloud Server. Section 5 includes the conclusion of our study paper.

II. Problem Formulation

The cloud servers are generally far away from end devices or mobile devices, therefore, the time to complete the requested tasks tends to be long due to the congestion, latency of the network, and queueing delay at the cloud server. In order to reduce the time to process the requested task, the edge layer serves the requests of mobile devices instead of the cloud server.

Vehicles will serve as edge servers in this paper although vehicles have limited computation and storage capability compared with cloud server. An edge gateway is another important component that intelligently classifies the applications to be processed on the edge or in the cloud depending on the requirements of computing resources. When a mobile device requires more computational resources to process a task, it sends the task to the nearest edge gateway. Then the edge gateway selects the cluster of edge servers that have the service of the task and chooses the optimal edge server in the selected cluster using Q-learning based dispatch algorithm.

In our proposed design, we assume that there are N number of services provided by the cloud server and n applications among those N are popular services (N>n), so n services may be processed by edge servers. We divide n services into K groups of services and pre-installed them to the public buses/taxis. For example, when there are 100 popular services and they are divided into 20 groups of services (service group 1, service group 2, . . . , service group 20), then every edge server has all 5 services of a service group among 20. We assume that service groups are 20 equally distributed and pre-installed into edge server vehicle.

The edge gateway can collect the information on which edge server (bus/taxi) has what kind of service group. After that, the edge gateway creates clusters of edge servers with the same service groups and within the same cellular area of the edge gateway. When a service request comes to the edge gateway, it selects the cluster that matches the service of the task. For every cluster, there will be a Deep Q Network run in the edge gateway to choose the best edge server to process the task. Then the edge gateway sends the task (service request) to the chosen edge server.

The overview architecture design is shown in Fig. 1. The edge gateway acts as the agent in DQN, which senses the state vectors from the edge devices.

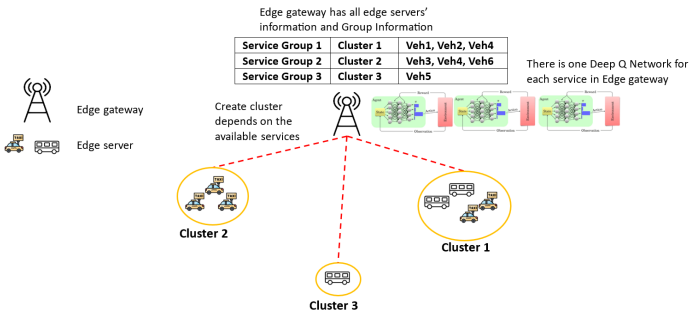

Fig. 1. Overview architecture of clustering the vehicles in Edge Computing

We consider public transportation (buses and taxis) as edge servers in this paper because they run at stable moving speeds and are evenly distributed in a cell area, moreover, they are also used for public purposes. It is easier and more consistent to collect the state vectors and decide the action vector for the edge gateway in Deep Q Network. We use the predicted future position of the edge servers in Deep Q Network. We can get that information from the fixed route information of the buses and the navigation system of taxis.

In the edge gateway, the state, action, and reward vectors for Deep Q Network of each service are defined as follows.

State vector: The edge gateway collects the state information as follows: 1. Workload of the requested task from mobile devices 2. Deadline of the task 3. Current workload of the edge servers (cluster members) 4. Current location and moving speed of the edge servers (cluster members) 5. Predicted future position of the edge servers (cluster members)

Action Vector: The edge gateway decides the action after analyzing the states of the environments. The action is to choose the two suitable (the best and the second best) edge servers in the cluster and dispatch the requested task.

Reward Vector: The edge gateway looks for the reward for the edge server if one of the chosen edge servers is not overloaded and both the requested mobile device and the chosen edge server are in the range of the edge gateway (after processing the task). If the action satisfies these conditions, the reward is gained. Otherwise, the penalty is imposed in DQN agent. As an example, a reward means that the edge server processed the tasks instead of the cloud server; thus, that edge server should be given the reward. In the real world, if the mobile user is subscribed to cloud resources or mobile services, the cloud or the mobile operator should give some percentage of the monthly rate to that edge server as a reward.

A. Creating the cluster of services in Edge Gateway

Edge gateway needs to collect the information of all edge servers to create clusters with edge servers that have the same service. Basically, the edge gateway has the information of all edge servers when they are in the range of the edge gateway, that is in the cell area of the base station. If there is only one member in the cluster, the edge gateway does not need to deploy or train the Deep Q Network, but if there are multiple

cluster members, the edge gateway needs to deploy and train the Deep Q Network for each service.

The workflow for creating the clusters depends on services in the edge gateway, as shown in Fig. 2.

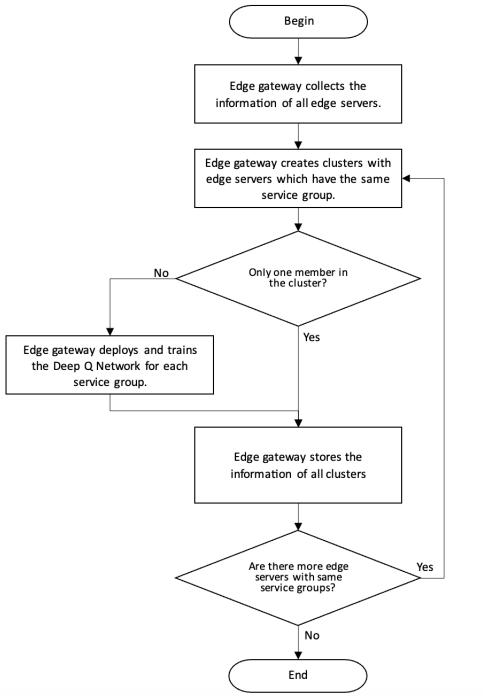

Fig. 2. Workflow for Creating the Clusters in Edge Gateway

B. Deep Q Network for each cluster in Edge Gateway

When a mobile device sends the task to the nearest edge gateway, the edge gateway selects the cluster that has the service of the task. Then, the edge gateway chooses the two optimal cluster members(edge servers) inside the cluster using the DQN network. The proposed Deep Q- Network algorithm in [12] is an efficient solution to offload the request optimally, which improves the overall performance of the network. According to [12], the authors used the DQN state vectors as the current workload of the edge server that has been assigned the service request, the number of tasks to be offloaded, the number of tasks remaining in the queue of the current edge server, and the future predicted position of the vehicles. In our proposed Deep Q Network, the edge gateway serves as a DQN agent, and the state vectors of DQN are already explained in detail in the previous page. As the DQN agent, the edge gateway utilizes these state vectors to find Q values and selects the action with the two highest predicted Q values.

$$a_{\mathrm{t}} = max(Q(s_{\mathrm{t}}, a_{\mathrm{t}})) + noise \tag{1}$$

The proposed technique efficiently utilizes the resources of the network and has better performance in case of noise in the network between the mobile device and the edge server that will perform the task. After choosing the edge server, the edge gateway calculates not only the reward vector but also the targeted Q value and loss function to get the updated parameter θ and update the parameter θ in the Q Neural Network and Target Neural Network. Targeted Q Value is defined as follows:

$$Q^{\mathrm{T}} = r_{\mathrm{t}} + \gamma max(Q^{1}(s_{\mathrm{t+1}}, a_{\mathrm{t}}), \ldots, (Q^{\mathrm{N}}(s_{\mathrm{t+1}}, a_{\mathrm{t}})) \tag{2}$$

The Loss Function uses the mean square error function between the targeted Q value and the predicted Q value.

$$L(\theta) = (Q^{\mathrm{T}} - Q^{\mathrm{P}})^2 \tag{3}$$

The workflow of DQN for each service in the edge gateway is constructed in Fig. 3 [12].

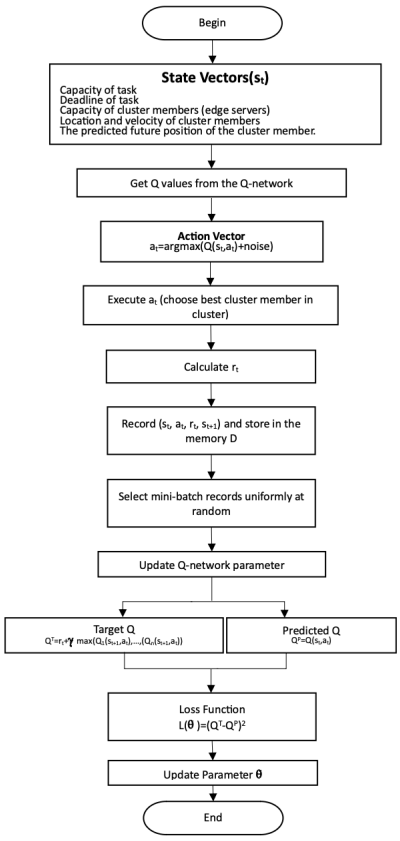

Fig. 3. Workflow of DQN for each service in Edge Gateway

C. Task Allocation at Edge Gateway

When a mobile device sends a service request to the edge gateway, the edge gateway looks for the cluster that has the service of the task. If there is no cluster with the service of the task, the edge gateway sends the task to the cloud server. If there is only one cluster member in the cluster, the edge gateway sends the task directly to that edge server.

Otherwise, the edge gateway chooses the two suitable edge servers using the DQN technique. The first chosen edge server is preferred to process the task, but if it is overloaded, i.e., if the CPU utilization of the chosen edge server is above a certain level, the second one will process the task. After processing the task, the edge server sends back the result of the task to

the mobile device through the edge gateway and informs the edge gateway to calculate the reward vector.

In the case that both of the two chosen edge servers using DQN are overloaded or there is only one edge server in the cluster and that edge server is overloaded, the edge gateway sends the task to the cloud server to process the task.

We define the workflow of task allocation at the edge gateway as in Fig. 4.

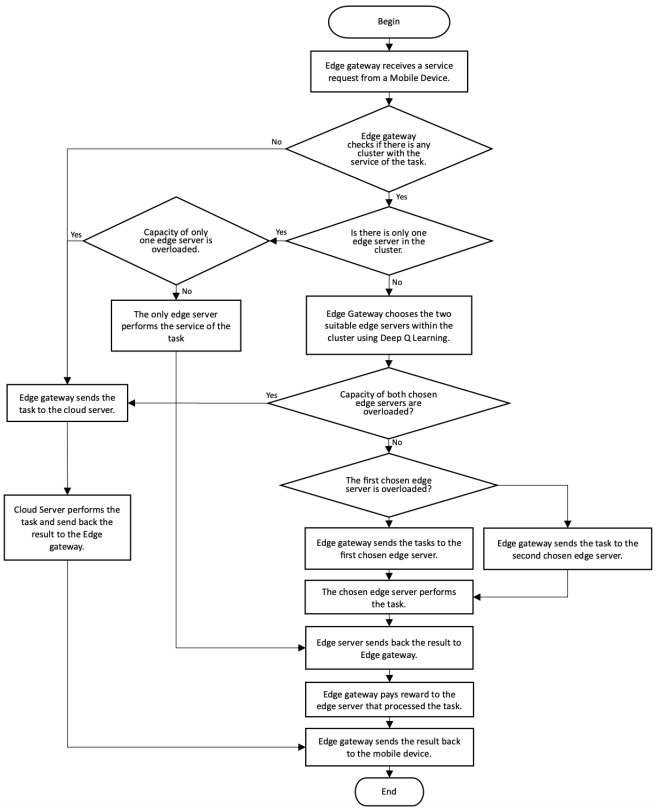

Fig. 4. Workflow of Task Allocation at Edge Gateway

III. Performance Models of Edge Computing

We compare the total processing time of a service when it is processed by an edge server with the time processed in a cloud server. Notations used in the performance models are defined as in the table. The data transfer time refers to the amount of time required to transfer a certain amount of data from one location to another and is calculated using the size of the data being transferred and the transfer speed or bandwidth [14]. In real-world situations, the actual data transfer time is also affected by propagation delay, representing the time taken for a signal to travel a specific distance in a medium at a given propagation speed [15]. Thus, the total data transfer time can be computed using

$$t_{\text{M,EG}} = (Data/dt_{\text{M,EG}}) + PD_{\text{M,EG}} \tag{4}$$

$$t_{\text{EG,CM}} = (Data/dt_{\text{EG,CM}}) + PD_{\text{EG,CM}} \tag{5}$$

$$t_{\text{EG,Cloud}} = (Data/dt_{\text{EG,Cloud}}) + PD_{\text{EG,Cloud}} \tag{6}$$

Notation	*Description*
t_e	The execution time to process the task in the server
t_{wc}	The waiting time in a queue of cloud server.
$t_{M,EG}$	The data transfer time between mobile device and edge gateway
$t_{EG,CM}$	The data transfer time between edge gateway and cluster member.
$t_{EG,Cloud}$	The data transfer time between the edge gateway and cloud server.
$t_{M,Cloud}$	The data transfer time between mobile device and cloud server
Data	Data size of the task.
$d_{tM,EG}$	Data transfer rate between mobile device and edge gateway.
$dt_{EG,Cloud}$	Data transfer rate between edge gateway and cloud server.
$dt_{EG,CM}$	Data transfer rate between edge gateway and cluster member.
$t_{M,EG}$	Time to transfer data from mobile device to edge gateway.
$t_{EG,Cloud}$	Time to transfer data from edge gateway to cloud server.
$t_{EG,CM}$	Time to transfer data from edge gateway to vehicle.
PD_{cloud}	Propagation Delay between cloud server and mobile device.
$PD_{M,CM}$	Propagation Delay between mobile device and cluster member.
$PD_{M,EG}$	Propagation Delay between mobile device and edge gateway.
$PD_{EG,CM}$	Propagation Delay between edge gateway and vehicle.
$PD_{EG,Cloud}$	Propagation Delay between edge gateway and cloud server.

In the 5G network, $dt_{M,EG}$ and $dt_{EG,CM}$ are the same. Thus, we denote dt_{EG} as

$$dt_{\text{M,EG}} = dt_{\text{EG,CM}} = dt_{\text{EG}} \tag{7}$$

For $PD_{EG,\ CM}$ and $PD_{EG,\ Cloud}$, the distance between the edge gateway and cloud server are usually longer than the distance between the edge gateway and edge server. However, the propagation speed between the edge gateway and the cloud server is much higher than the speed between the edge gateway and the edge server because the transmission medium is usually optical fiber between the edge gateway and the cloud server. Then we can assume that $PD_{EG,\ Cloud}$ and $PD_{M,\ EG}$ are almost the same. We also assume that $PD_{M,EG}$ and $PD_{EG,CM}$ are the same.

$$PD_{\text{M,EG}} = PD_{\text{EG,CM}} = PD_{\text{EG,Cloud}} = PD \tag{8}$$

Thus, the data transfer time from the mobile device to the edge server (cluster member) can be written as

$$t_{M,CM} = t_{M,EG} + t_{EG,CM} = 2(\text{Data}/\ dt_{EG}) + 2PD \tag{9}$$

Furthermore, the data transfer time from mobile to a cloud server can be written as

$$\begin{aligned} t_{M,Cloud} &= t_{M,EG} + t_{EG,Cloud} \\ &= (\text{Data}/\ dt_{EG}) + (\text{Data}/\ dt_{EG,Cloud}) + 2PD \end{aligned} \tag{10}$$

In the cloud server or the edge server, the request may wait in a queue after it arrives at the server until it is processed. Still, the waiting time at the edge server is much less than at the cloud server due to the cloud server being accessed by many clients at the same time. Thus, we will assume that the waiting time at the edge server is negligible.

According to Little's Law, the waiting time at a queue in the cloud server can be given by

$$t_{wc} = \rho/(\mu(1-\rho)) \tag{11}$$

where ρ = the traffic intensity λ/μ, μ = service rate at a cloud server, λ = arrival rate at a cloud server

We will assume that the data size of the requested task and

the result data of the requested task after processing are the same.

When using an edge server, it takes additional time to select an appropriate edge server using Deep Q Network. Thus, the total processing time of the task at the chosen edge server can be written in the form of

$$t_{\text{totalE}} = t_e + 2t_{\text{M,CM}} = t_e + 2(2(Data/dt_{\text{EG}}) + 2PD) + t_{\text{DeepQ}} \quad (12)$$

The total processing time of the task on the cloud server can also be defined as

$$\begin{aligned} t_{\text{totalC}} &= t_e + t_{wc} + t_{\text{M,Cloud}} \\ &= t_e + t_{wc} + 2((\text{Data}/\ dt_{\text{EG}}) + (\text{Data}/\ dt_{\text{EG,Cloud}}) + 2\text{PD}) \end{aligned} \quad (13)$$

IV. Numerical Results

In 5G mobile environment, the data transfer speed is higher than 1Gbps (Gigabits per second). And the fastest-ever 5G data transmission rate in a stationary environment is 7.5 Gbps [16]. Thus, we regard the average data transfer rate of 5G environment as 3Gbps.

$$dt_{\text{EG}} = 3\text{Gbps} = 375\ \text{MBps (Mega Byte per second)}$$

Some common assumptions for data transfer rates in a fiber optic connection between a base station and a cloud server could be:

1. Moderate assumption: A data transfer rate of 1 Gbps. This assumes a reasonably high-speed fiber optic connection that can handle 1 billion bits of data per second.
2. High-end assumption: A data transfer rate of 10 Gbps. This assumes a high-capacity fiber optic connection capable of transferring 10 billion bits of data per second.

Thus, we will define the average data rate from the base station to the cloud server as 5 Gbps.

$$dt_{\text{EG,Cloud}} = 5\ \text{Gbps} = 625\ \text{MBps}$$

According to our simulation result, the time to select an edge server using Q Network is 0.004 second.

$$t_{\text{DeepQ}} = 0.004\ \text{second}$$

Here, we will assume the execution time as 1 second and the propagation delay PD as 0.001 seconds in both the cloud server and the chosen edge server.

Fig. 5 illustrates the relation between the arrival rate at a server and the waiting time at the queue of the server according to (11). The more the arrival rate (request rate) increases, the longer the waiting time increases when the service rate μ is 122. Thus, in the case of cloud server, there are many clients accessing the cloud server at the same time in real life and the more clients access the cloud server, the longer the waiting time according to the number of accesses.

A. Case 1 analysis

We compare the total time to process a task at the cloud server and an edge server according to the data size of the task with the value t_{wc} fixed.

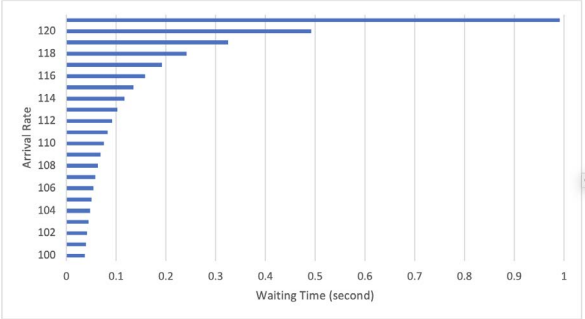

Fig. 5. Waiting Time depends on arrival rate λ

We set the average arrival rate as 120 requests per second and the average service rate as 122 at a cloud server. Then, the waiting time at a cloud server according to (9) will be as follows. This is almost the same as the value shown in Fig. 5.

t_e	t_{wc}	PD
1 second	0.4918 second	0.001 second

TABLE I. Initial Parameters for Figure 6

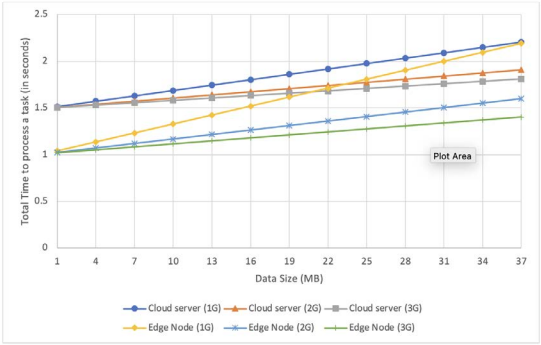

Fig. 6. Total Processing Time at the cloud server and cluster member according to Data Size

Fig. 6 shows the total turnaround time of service by the cloud server and by an edge server concerning the variable data size of the task using the parameters of Table 1. Turnaround time, equivalent to the total processing time, is the duration between the service request made by a mobile device and the arrival of the service result back to the device. In Fig. 6, we observe that the turnaround times for the cloud server are consistently higher compared to those of the edge server. As the data size increases, the total processing time gap narrows, particularly at lower data transfer rates like 1Gbps or 2Gbps in a 5G environment. Thus, processing the task at the edge server is better than the cloud server when the requested task size is not too big. However, when the data transfer rate is 3Gbps, the turnaround time of the task in the edge server is always better than the cloud server as we can see from both lines of the edge server (3G) and cloud server (3G). These days, the data transfer rate of 5G environment is getting higher, so if

the data transfer rate exceeds 3Gbps, the processing time at the edge server will be superior to that of the cloud server.

B. Case 2 analysis

In this experiment, we compare the total time to process a task at a cloud server and cluster member concerning the arrival task at the cloud server and the fixed data size of the task.

Data Size	t_e	*PD*
15 MB	1 second	0.002 second

TABLE II. Initial Parameters for Figure 7

The total time to process that task at the cloud server will be:

$$t_{totalC} = 1 + t_{wc} + 2((15/\ 125) + (15/625)) + 0.004$$

Fig. 7 compares the total turnaround time of the cloud server and the edge server with respect to the arrival rate (request rate) at the cloud server. As the arrival rate increases, the total turnaround time at the cloud server also increases significantly. At a lower arrival rate, the processing time at the edge server may be higher than that at the cloud server but as the arrival rate grows, the turnaround time of the edge server is better than the cloud server. This is because the waiting time at the cloud server increases as the arrival rate of requests becomes higher. Though using the edge server when the wireless communication speed is 1Gbps is not advantageous in turnaround time, it shows similar or better performance than the cloud server when the wireless communication speed is 2 and 3Gbps.

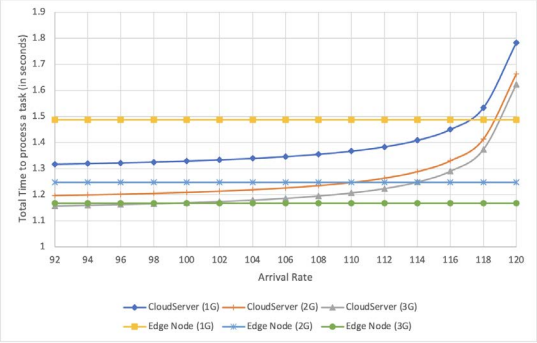

Fig. 7. Total Processing Time at the cloud server and cluster member according to Arrival Rate at the cloud server

V. Conclusion

This paper showed the architecture of the vehicular edge computing system, where the edge servers are managed by the edge gateway in the base station. And the proposed DQN algorithm is an efficient solution to distribute the workload among the edge nodes (vehicles), which improves the overall performance within the cluster. We also modeled the time to process a service request from a mobile user. We compared the total processing time of the task requested to the edge server with the time of the cloud server.

According to the Case 1 analysis, we can notice that if the data size is bigger than 40 MB during the 1Gbps data transfer rate or 103 MB during the 2Gbps data transfer rate, the processing of the task on the edge server is worse than on the cloud server. But when the data transfer rate is over 3Gbps, the processing on the edge server is almost always better than on the cloud server since we can see that the lines in the graph for the data transfer rate of 3Gbps are parallel to each other. Furthermore, we found out that the more the arrival rate increases at the cloud server, the processing time at the cloud server becomes larger and larger according to the case 2 analysis.

VI. Acknowledgement

This results was supported by Regional Innovation Strategy (RIS) through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE)(2021RIS-004).

References

[1] Abdelhamid, Sherin, Hossam S. Hassanein, and Glen Takahara. "Vehicle as a resource (VaaR)." IEEE Network 29.1 (2015): 12-17.
[2] Bitam, Salim, Abdelhamid Mellouk, and Sherali Zeadally. "VANET-cloud: a generic cloud computing model for vehicular Ad Hoc networks." IEEE Wireless Communications 22.1 (2015): 96-102.
[3] Jang, Insun, et al. "The software-defined vehicular cloud: A new level of sharing the road." ieee vehicular technology magazine12.2 (2017): 78-88.
[4] Sheikh, Hafiz Fahad, et al. "Energy-and performance-aware scheduling of tasks on parallel and distributed systems." ACM Journal on Emerging Technologies in Computing Systems (JETC) 8.4 (2012): 1-37.
[5] Toor, Asfa, et al. "Energy efficient edge-of-things." EURASIP Journal on Wireless Communications and Networking 2019.1 (2019): 1-11.
[6] Hou, Xueshi, et al. "Vehicular fog computing: A viewpoint of vehicles as the infrastructures." IEEE Transactions on Vehicular Technology 65.6 (2016): 3860-3873.
[7] Chiang, Mung, and Tao Zhang. "Fog and IoT: An overview of research opportunities." IEEE Internet of things journal 3.6 (2016): 854-864.
[8] Dastjerdi, Amir Vahid, and Rajkumar Buyya. "Fog computing: Helping the Internet of Things realize its potential." Computer 49.8 (2016): 112-116.
[9] Bonomi, Flavio, et al. "Fog computing and its role in the internet of things." Proceedings of the first edition of the MCC workshop on Mobile cloud computing. 2012.
[10] Lee, Seung-seob, and SuKyoung Lee. "Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information." IEEE Internet of Things Journal 7.10 (2020): 10450-10464.
[11] Hou, Xueshi, et al. "Vehicular fog computing: A viewpoint of vehicles as the infrastructures." IEEE Transactions on Vehicular Technology 65.6 (2016): 3860-3873.
[12] Maan, Ujjawal, and Yogesh Chaba. "Deep Q-network based fog node offloading strategy for 5 G vehicular Adhoc Network." Ad Hoc Networks 120 (2021): 102565.
[13] Hameed, Ahmad Raza, et al. "Energy-and performance-aware load-balancing in vehicular fog computing." Sustainable Computing: Informatics and Systems 30 (2021): 100454.
[14] Forouzan, B. A., and S. C. Fegan. "Data Communication and Networking. McGraw-Hill Companies." Inc. New York (2007).
[15] Balch, Mark. Complete digital design: a comprehensive guide to digital electronics and computer system architecture. McGraw-Hill Education, 2003.
[16] Gozalvez, Javier. "Samsung Electronics Sets 5G Speed Record at 7.5 Gbs [Mobile Radio]." IEEE Vehicular Technology Magazine 10.1 (2015): 12-16.