Workload Processing Method Considering Priority Based on Hot Standby System in RSU

Jinwon Jeong[1], Sejin Kim[1], Joonmin Gil[2], Kwangsik Chung[3], Heonchang Yu[1]

[1] Department of Computer Science and Engineering, Korea University, Seoul, 02841, Korea
[2] Department of Computer Engineering, Catholic University of Daegu, Daegu, 38430, Korea
[3] Department of Computer Science, Korea National Open University, Seoul, 03087, Korea
{jin4812, sejjj120, yuhc}@korea.ac.kr[1], jmgil@cu.ac.kr[2], Kchung0825@knou.ac.kr[3]

Abstract. In a connected vehicle environment, if the operation of the mobile edge server is not guaranteed, the workload must be offloaded to RSU to continuously maintain real-time sensor data processing. However, if an efficient resource management is not performed within the RSU, which has a poor computing resource environment, the safety of the vehicle may not be guaranteed because urgent information cannot be received. Therefore, in this paper, we propose a method to provide RSU service more stably by constructing a Hot Standby system in RSU and conducting research to quickly process workload based on priority.

Keywords: RSU, OBU, MEU, Container, Hot Standby System, Priority

1 Introduction

The main function of RSU (Road-Side Unit) is to facilitate communication among vehicles, transportation infrastructure and other devices by transmitting data through DSRC (Dedicated Short-Range Communication) technology according to industry standards. RSU acquires necessary traffic information such as time, speed, and vehicle location, and communicates with each other with the OBU (On-Board Unit) of the driving vehicle using DSRC. Therefore, it is possible to prevent collisions caused by traffic congestion and accidents of nearby vehicles through RSU. However, since RSU has poor computing resources compared to data centers, there may be a delay due to resource contention [1]. In other words, it may not be possible to deliver the most urgent information to the OBU among the indiscriminate large amounts of data entering the RSU. This can lead to additional accidents by impairing the safety of the vehicle as it is not accurately recognized the situation around the road [2]. Therefore, in this paper, we propose a method to minimize the delay time that occurs when processing workloads by building a hot standby system in RSU and to provide efficient and stable services by adjusting resource allocation based on priority within the limited computing resources of RSU.

2 Related Work

In the existing research, a study was conducted to keep real-time sensor data processing by offloading the workload to RSU [3] when work in the smartphone-based edge server developed to support fast response time and low network traffic to provide services such as safety information and accident prevention for connected vehicle is not guaranteed. Besides, the environment is configured as shown in Figure 1 to indicate a situation where communication between the OBU attached in the vehicle and the Mobile Edge Unit (MEU) equipped with the edge server function on the smartphone is impossible. The OBU consists of a Switcher and a Forwarder, and the Switcher communicates with the Generator that plays the role of OBDⅡ (On-Board Diagnostics) about connected vehicle sensor data and is in charge of transmitting and receiving data with the MEU.

When the RSU receives vehicle information and sensor data from the OBU on behalf of the MEU, the sensor data is transmitted to the RSU Manager through the internal Forwarder. In the RSU Manager, a container is created and a workload is processed in it to isolate the resources required to process sensor data received from the OBU and to guarantee data processing performance. When the condition of the vehicle and whether there is an accident is determined in the container, the analyzed information is transmitted to the RSU Manager, and if it is determined that the vehicle's condition is abnormal or an accident has occurred, it is also transmitted to the cloud server. The Forwarder receives the analyzed information from the RSU Manager and broadcasts it to other nearby OBUs.

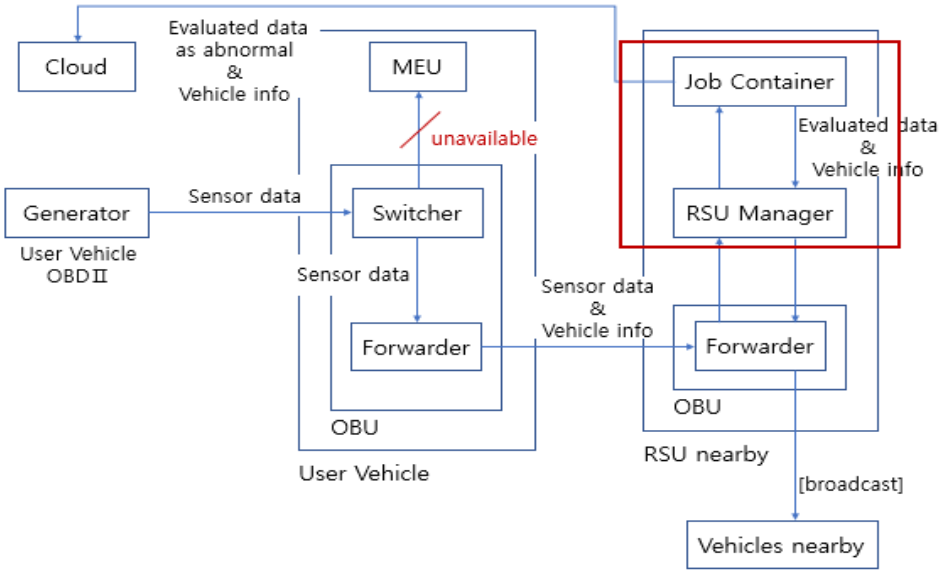

Fig. 1. Existing Configuration of the RSU

However, since the RSU system proposed in the previous research creates a container whenever it receives the first sensor data from a nearby vehicle, this may cause an initial delay time. That is, there may be a problem in that the final delay time between sending the first vehicle sensor data to the RSU and receiving the analyzed sensor data information increases. Therefore, instead of the Cold Standby system that creates a container every time a new request comes in, we propose a method to process the workload in the Hot Standby system that directly offloads the workload for the vehicle by creating a specific number of containers in advance. Besides, if CPU contention occurs while processing a large workload in RSU, we propose a priority-based resource scheduling method to quickly and reliably process the workload offloaded to the container with high priority by adjusting the CPU share of the container.

3 Proposed Systems

Hot Standby system enables continuous service provision by continuing to operate without system interruption even if one or more nodes fail. Applying the Hot Standby system to RSU can reduce the risk of unexpected errors that can significantly affect vehicle safety by operating without system downtime due to container creations. When creating a container, we can control the container's resource usage with various options. If no options are entered, by default all resources on the host will be available without restriction, which may lead to an imbalance in resource usage when running multiple containers. The container's CPU-shares option allows us to set how much CPU the container will occupy when there is contention for the CPU. In other words, by setting weight on a container, we can set how much CPU the container can use relatively. Therefore, it is possible to expect stable RSU services by increasing the CPU share of containers that need to process emergency data and it can process workloads in a short time providing emergency services.

3.1 Design

If the MEU determines that real-time sensor data processing is not possible, the workload is offloaded to the RSU. When the RSU Manager receives vehicle information and sensor data through the Forwarder, one of the pre-created containers in the RSU Server with sufficient resources processes the workload. When the container processes the workload and determines the status of the vehicle and whether there is an accident, the analyzed information is delivered to the RSU Manager. The Forwarder receives the analyzed information from the RSU Manager and broadcasts it to other nearby OBUs. When the RSU receives the second sensor data from the vehicle, the container of the vehicle is still maintained as it is, enabling continuous data transmission and reception. However, if the vehicle sensor data no longer comes in RSU within the time-limited, the container is destroyed.

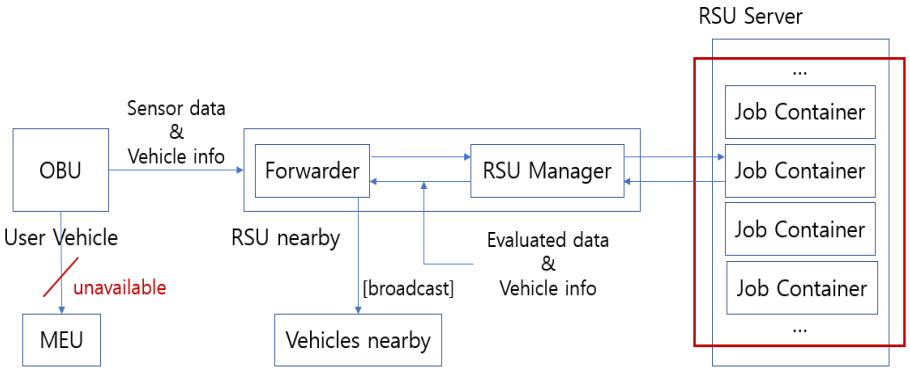

Fig. 2. Proposed Configuration of the RSU

3.2 Resource Scheduling Method Based on Priority

In this section, we propose a priority-based resource scheduling method for efficient resource management of containers in RSU. The terms required for algorithm description and explanation of each term are shown in Table 1. RSU has very limited computing resources, so resources must be carefully distributed and managed to each container. Besides, in order to reliably and quickly process emergency data, it is necessary to automate the scaling of container resources according to the priority of adjusting the resources of the container.

When communication between the OBU and the MEU is impossible, in order to offload the workload to the container in the RSU, the CPU and memory resource usage required for offloading are compared with the remaining resources of the container to determine whether to offload. If the container has enough resources, it proceeds offloading and processes the workload. If there are not enough resources, it tries offloading to another container. If the result of sensor data analyzed in the container is determined to be normal data, the container is given weight according to the relative order in which individual workloads are offloaded to each container. After that, priorities for appropriate resource adjustment of the container are assigned. The faster the offloaded order is than other containers, the larger the value for the corresponding weight. If abnormal data is detected in the results of the sensor data analyzed in the container, the weight assigned to the container according to the relative order of offloading and weight assigned only when abnormal data is detected are additionally assigned. Therefore, it is assigned a higher priority than the container that derived the result determined as normal data. When prioritized, the CPU share of each container is adjusted according to priority through the CPU-shares option. Therefore, through this scheduling method, it is possible to expect an RSU service that increases the CPU share of the container that needs to continuously process emergency data in a situation where CPU contention occurs and processes the workload in a short time.

Table 1. Terms and Description

Term	Description
VW_n	Workload of the vehicle node n
C_{ij}	A CPU utilization needed to offload the i^{th} *VW* to j^{th} container
RC_j	The remaining CPU utilization of the j^{th} container
M_{ij}	A memory utilization needed to offload the i^{th} *VW* to j^{th} container
RM_j	The remaining memory utilization of the j^{th} container
U_{ij}	Decide whether to offload the i^{th} *VW* to the j^{th} container
A_{ij}	The analysis result of i^{th} *VW* offloaded to j^{th} container
P_{ij}	The priority value of j^{th} container where i^{th} *VW* offloaded
α_{ij}	A weight value based on the relative order where i^{th} *VW* offloaded to j^{th} container (The faster the order, the larger the value)
β_{ij}	A weight value assign when A_{ij} is abnormal

Container Resource Scheduling Algorithm

```
if C_ij < RC_j and M_ij < RM_j
        U_ij ← 1
else if C_ij > RC_j and M_ij > RM_j
        U_ij ← 0
end if
if U_ij == 1
        run container's workload
        A_ij ← i^th Vehicle node's state
else if U_ij == 0
        attempt to offload to another container
end if
if A_ij is normal
        P_ij ← α_ij
        j^th container's CPU share *= P_ij
else if A_ij is abnormal
        P_ij ← α_ij * β_ij
        j^th container's CPU share *= P_ij
end if
```

4 Experiments

In this experiment, the delay time between sending vehicle sensor data to RSU and receiving the analyzed sensor data is measured and compared in the existing Cold Standby system and the proposed Hot Standby system. Besides, we apply a priority-based resource scheduling method to adjust CPU shares according to container priority, then compare the measured workload processing time of each container and validate the proposed method.

4.1 Experimental Setup and Procedures

For this experiment, as shown in Figure 3, an environment that can communicate and process sensor data generated in a connected vehicle environment in real-time was configured. The Generator is a program developed in C# language that sends vehicle sensor data to OBU using actual vehicle sensor data, and the collected vehicle sensor data is composed of CSV files. The environment of RSU and OBU is shown in Table 2.

Table 2. Experimental Environment

	RSU	OBU
MODEL	OptiPlex 3070	ETF-DO-02
CPU	i7-9700 CPU @ 3.00GHz x 8	Quad ARM Cortex-A53
RAM	32GB	4GB
OS	Ubuntu 18.04.4 LTS	Linux 4.9.58

When the vehicle's sensor data is generated by the Generator, the sensor data is sent to OBU1 via a web socket communication. OBU1 is unable to communicate with the MEU, so it uses DSRC communication to communicate vehicle information and sensor data to OBU2. And then OBU2 finally delivers this information and data to the RSU via a web socket communication.

Fig. 3. Experimental Configuration between RSU and OBU

To compare the delay time until receiving a response from the RSU in two Standby systems, it is assumed that there are 5 to 50 vehicle nodes around the RSU. For example, if there are 5 vehicle nodes, the Generator transmits the sensor data of 5 vehicle nodes separated by vehicle number to RSU at the same time. Whenever RSU receives sensor data, RSU analyzes and broadcasts it. Here, the delay time until one vehicle node receives the analyzed sensor data of 5 vehicle nodes including itself from the RSU is

all measured, and the difference between the minimum delay time and the maximum delay time is calculated. After performing this in each of the two Standby systems, the difference value of the derived delay time is compared, and the above process is performed according to the number of classified vehicle nodes.

The comparison of the workload processing time according to the priority-based CPU share of the container was performed for 5 vehicle nodes, and only one of the 5 vehicle nodes was set to be analyzed abnormal data in RSU. Besides, to indicate the situation where CPU contention occurs in RSU, the stress tool [4] was used to load the CPU with 10 threads in each pre-created container. Assuming that each vehicle node continues to transmit sensor data to the RSU at the same time, we measure the time that each container has processed the workload after the CPU share has been adjusted according to the priority of each container. After that, the values of the workload processing time measured continuously in each container are compared, separated by the number of times the workload is processed.

4.2 Experimental Results and Performance Analysis

Figure 4 shows the difference between the minimum delay time and the maximum delay time after receiving all vehicle sensor data analyzed as many as the number of vehicle nodes around the RSU in Cold Standby and Hot Standby systems. As the number of vehicle nodes around the RSU increases, the difference in delay time tends to increase in the existing Cold Standby system, but in the proposed Hot Standby system, the change in the difference of delay time is very small. This shows the result of reducing the delay time by not creating a container every time RSU receives the vehicle's sensor data but creating a container in advance so that the workload is processed immediately.

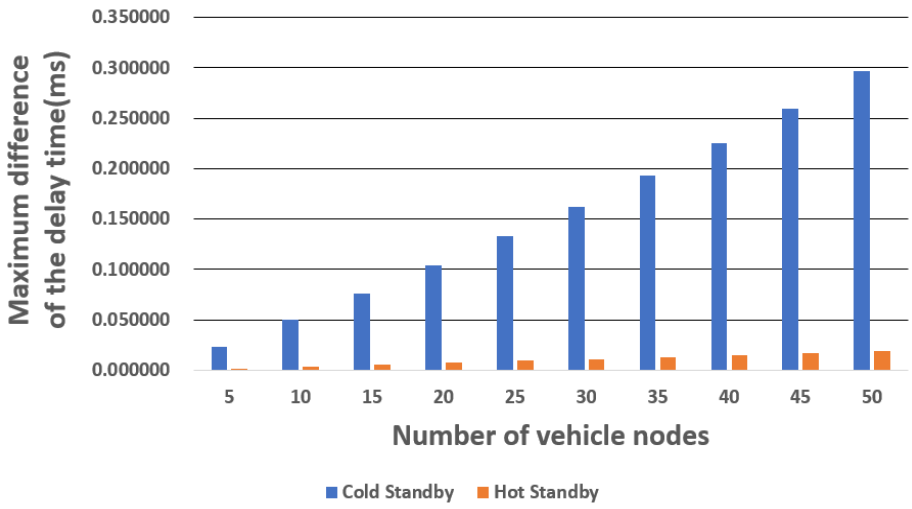

Fig. 4. Comparison of Delay Time in Cold Standby System and Hot Standby System

Comparing the workload processing time according to the priority-based CPU share of the container is shown in Figure 5. While the CPU is kept busy, the initial CPU share of the five containers processing offloaded workloads is maintained at the same rate. Then, abnormal data is detected among sensor data analyzed in the third container and CPU shares are adjusted according to the priority-based resource scheduling method. At this point, the CPU share increased by about tenfold compared to other containers. As the CPU share of the third container increases, it can be seen that the container has a shorter workload processing time than other lower priority containers, as shown in Figure 5. This shows that in a situation where CPU contention occurs in the RSU, the CPU share of the container that needs to process urgent data is increased and the workload is processed in a faster time.

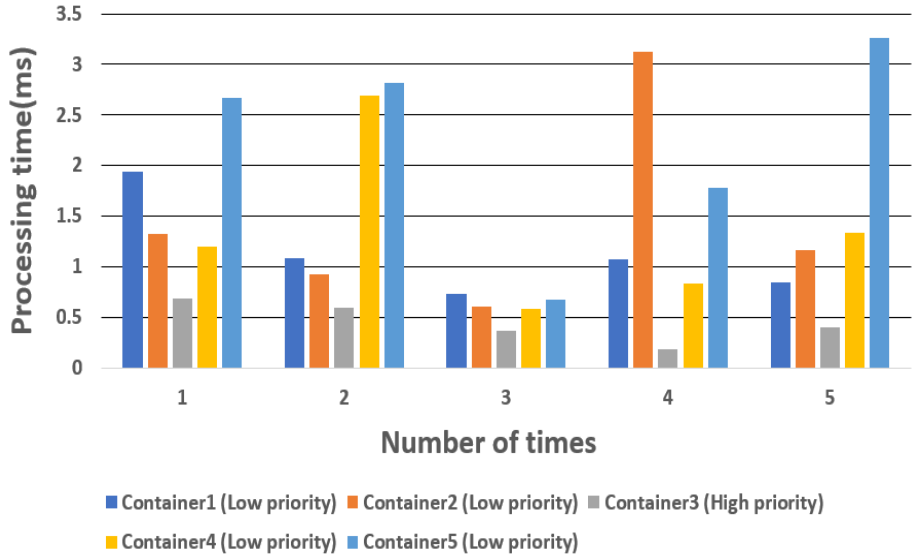

Fig. 5. Comparison of Workload Processing Time Based on Priority

5 Conclusions and Future Work

In this paper, by constructing a Hot Standby system in RSU, which provides various traffic information services, we proposed a method to minimize the delay time that occurs when processing the workload. Besides, a method to provide efficient and stable services by adjusting resource allocation according to priority within the limited computing resources of RSU was proposed. As a result of applying the proposed Hot Standby system and priority-based resource scheduling method, the delay time until receiving the analyzed data from the RSU was reduced and emergency data could be stably processed. Therefore, it is possible to expect a service that guarantees the safety of the vehicle by preventing a dangerous situation that may be encountered due to the failure to receive emergency services quickly.

However, in the method proposed in this paper, there is a limitation that performance overhead may occur due to idle resources. It is because some containers are not used among the pre-created containers. As a future study, we plan to study a method for the creation of prediction-based dynamic containers to prevent the waste of computing resources of RSU.

Acknowledgement

1. This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2018-0-01405) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation).
2. This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00480, Developing the edge cloud platform for the real-time services based on the mobility of connected cars).

References

1. V. V. Paranthaman, Y. Kirsal, G. Mapp, P. Shah, H. X. Nguyen, "Exploiting resource contention in highly mobile environments and its application to vehicular ad-hoc networks", IEEE Trans. Veh. Technol., vol. 68, no. 4, pp. 3805-3819, Apr. 2019.
2. M. Fogue, J. Sanguesa, F. Martinez and J. Marquez-Barja, "Improving roadside unit deployment in vehicular networks by exploiting genetic algorithms", Appl. Sci., vol. 8, pp. 86, Jan. 2018.
3. J. W. Jeong, J. H. Lee, J. M. Gil, H. C. Yu, "Workload Processing Method using Containers in RSU", KINGPC, Aug. 2020.
4. https://linux.die.net/man/1/stress