Proceedings of the SICE Annual Conference 2016
Tsukuba, Japan, September 20-23, 2016

Vehicle Control System Coordinated Between Cloud and Mobile Edge Computing

Kengo Sasaki[1,2†], Naoya Suzuki[1], Satoshi Makido [1] and Akihiro Nakao[2]

[1]TOYOTA CENTRAL R&D LABS., INC., Aichi, Japan
(Tel: +81-561-71-7440; E-mail: sasaki-ken@mosk.tytlabs.co.jp)
[2]University of Tokyo, Tokyo, Japan
(Tel: +81-3-5841-2664; E-mail: nakao@nakao-lab.org)

Abstract: One of the challenges in autonomous driving is limited sensing from a single vehicle that causes spurious warnings and dead-lock situations. We posit that *cloud-based vehicle control system*[1] is promising when a number of vehicles must be controlled, since we can collect information from sensors across multiple vehicles for coordination. However, since cloud based control has inherent challenge in long-haul communication susceptible to prolonged latency and packet loss caused by congestion, mobile edge computing (MEC)[2] recently attracts attention in ITS in the next generation mobile network such as 5G. Although edge servers can perform data processing from the vehicles in ultra low latency in MEC, computational resources at edge servers are limited compared to cloud. Therefore, dynamic resource allocation and coordination between edge and cloud servers are necessary. In this paper, we propose *infrastructure-based vehicle control system* that shares internal states between edge and cloud servers, dynamically allocates computational resources and switches necessary computation on collected sensors according to network conditions in order to achieve safe driving. We implement a prototype system using micro-cars and evaluate the stability of infrastructure-based vehicle control. We show that proposed system mitigates instability of cloud control caused by latency fluctuation. As a result, when controlled from the cloud with 150ms latency, micro-cars deviate by over 0.095m from the course for the 40% of the entire trajectory possibly causing car accidents. On the other hand, MEC-based control stabilizes the driving trajectory. Also, our proposed system automatically switches control from cloud and from edge server according to the network condition without degrading the stability in driving trajectory. Even when the ratio of time of control by edge server to that by cloud is suppressed to 54%, we can achieve almost the same stability as in full control by edge controller.

Keywords: Autonomous Driving, Network Virtualization, Mobile-Edge Computing

1. INTRODUCTION

Recently, various research has been focused on developing *autonomous vehicles*[3, 4]. They autonomously plan their driving trajectory and control themselves according to the sensors, such as lidar, camera, GPS, and so on. Although such self-contained control systems are attractive, they also have disadvantages. First, their sensing areas are limited, e.g. they cannot sense behind obstacles. Second, they always have the possibility of deadlock. If two or more vehicles encounter at an intersection, they cannot decide which vehicle should pass through first. At last, the cost of the vehicle equipped with rich sensors are expensive.

To overcome these challenges, we have proposed *cloud-based vehicle control system*[1] that periodically aggregates such sensor information as positions, velocities, and so on, from multiple vehicles in vicinity and controls them from the cloud. It not only avoids deadlock but also frees vehicles from costly rich sensors. However, the cloud based approach introduces the other problems caused by the Internet. How to compensate for the latency is one of the important problems for stable control. Latency varies according to the time of day. Furthermore, due to network congestion [1] , the Internet has suddenly large latency and interferes with the stable remote control of the vehicle.

Mobile-Edge Computing (MEC)[2] recently attracts attentions as one of the core mobile technologies[5]. MEC installs computational nodes referred to as edge servers at the edge of the network. If the edge server is located at the mobile network edge, i.e. at the base station, it is referred to as E-UTRAN Node B(eNB). In case at the edge of the Internet, it is referred to as Packet data network GateWay (P-GW). The edge server can process the data from vehicles with low latency. Although using the edge server is attractive from a network point of view, the computational resources of the edge servers are bound to be limited compared to the considering the scale of the network in general.

In this paper, we propose *infrastructure-based vehicle control system* that uses the edge server as a computational node together with the cloud-based vehicle control system as shown in Fig. 1. Both cloud and edge servers have vehicle controllers and are referred to as the cloud controller and the edge controller. The edge controller can achieve stable control of vehicles against the network fluctuations. To balance computational load between edge and cloud servers, proposed system has an automatic switching method of vehicle controller between edge and cloud servers, which shares internal states between edge and cloud servers to prevent unstable control by automatic switching. Since the latency is found the

† Kengo Sasaki is the presenter of this paper.
[1]Network congestion is the reduced quality of service that occurs when a network node is carrying more data than it can handle.

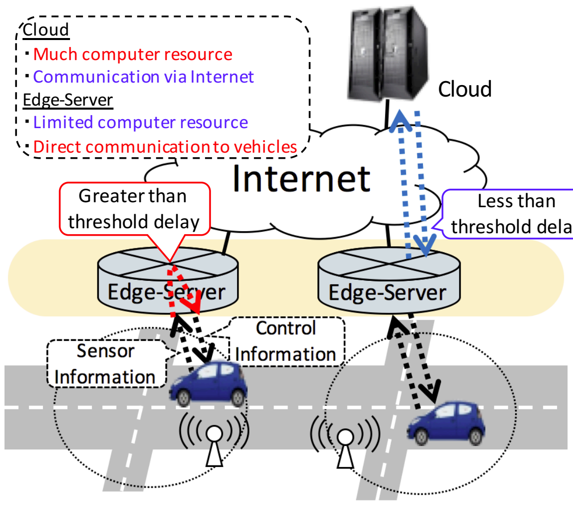

Cloud
・Much computer resource
・Communication via Internet
Edge-Server
・Limited computer resource
・Direct communication to vehicles
Cloud
Internet
Greater than threshold delay
Less than threshold delay
Edge-Server
Edge-Server
Control Information
Sensor Information

Fig. 1 Infrastructure-based vehicle control system

most crucial metric to reduce in our previous work[1], our system monitors latency between edge and cloud servers and switches control between them as follows. If the latency is less than threshold latency, the edge server forwards sensor information from vehicles to the cloud server and the vehicle controller is switched to the cloud server; otherwise, the edge server stops forwarding the sensor information and takes over the vehicle control.

To the best of our knowledge, our work in this paper is the first to propose the coordinated vehicle control between cloud and MEC. Although applications of MEC to vehicles have been proposed in [6], they are limited to entertainment services and hazard warnings and vehicle control has not been fully explored. Although the vehicle control systems through cloud[7, 8] have been proposed, they assume rich sensors and use cloud for assisting them.

We implement the prototype using micro-cars and evaluate the stability of vehicle control. To balance the computational load between edge and cloud servers, we additionally evaluate the control ratio and the amount of traffic.

The primary contributions of this paper are three fold. First, when controlled from the cloud with 150ms latency, micro-cars deviate by over 0.095m from the course for the 40% of the entire trajectory possibly causing car accidents. On the other hand, MEC-based control stabilizes the driving trajectory. Second, our proposed system automatically switches control between the edge and the cloud servers according to the network condition without degrading the stability in driving trajectory. Even when the ratio of time of control by edge server to that by cloud is suppressed to 54%, we can achieve almost the same stability as in full control by edge controller. At last, we identify the trade-off between quick response to change of network condition and the amount of traffic between edge and cloud servers.

This paper is organized as follows. In Section 2, proposed system and the architecture are presented. In Section 3, prototype using the micro-car is presented. Section 4 measure and evaluate the proposed system. Section 5 briefly concludes.

2. PROPOSED SYSTEM

In this section, to save computational resource usage of the edge servers and avoid network problem, we propose Infrastructure-based vehicle control system which has automatic switching method of vehicle controller between edge and cloud servers.

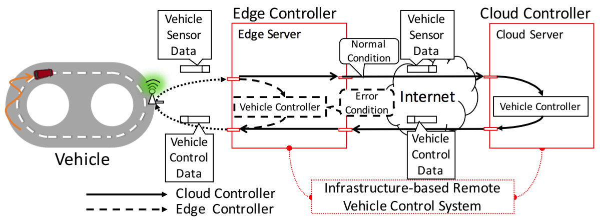


Fig. 2 Architecture of the proposed system

Fig. 2 shows the architecture of the proposed system. The proposed system consists of edge and cloud controllers. Both cloud and edge controllers have vehicle controllers with the same capabilities and either of them controls the vehicles. The edge controller monitors network condition between edge and cloud servers and switches control between them.

2.1. Cloud controller

Vehicles periodically send the "Vehicle Sensor Data (VSD)" packets with vehicle ID, position, velocity and yaw, receives "Vehicle Control Data (VCD)" packets with vehicle ID, steering angle, accelerator value and brake value and set handle, accelerator and brake in accordance with the VCD packets.

In case the cloud controller controls the vehicles (solid line in Fig. 2), the edge controller forwards VSD packets from the vehicle to the cloud and VCD packets from the cloud to the vehicle. The cloud controller receives VSD packets and sends VCD packets. The cloud controller can aggregate wide range of VCD packets and use a lot of computational resource for remote control system. However, it introduces various network problems, such as latency, packet loss, network congestion and so on, caused by the Internet.

2.2. Edge controller

In case the edge controller controls the vehicles (dotted line in Fig. 2), the edge controller receives VSD packets and sends VCD packets without the cloud. We assume that the edge server is deployed in the access network and the edge server needs to process vehicle control with limited computational resource. On the other hand, the edge server can execute vehicle control without network problem caused by the Internet.

2.3. Automatic switching method

In our proposed automatic switching method, the edge controller monitors the latency between edge and cloud controllers and calculates the average latency L_{ave}. The latency is measured between the transmission of a VSD packet and the reception of a VCD packet. The L_{ave} is calculated using simple moving average, i.e., if the windows size of L_{ave} is N_t and the previous N_t latencies are $t_1, t_2, ...t_{N_t}$ then the formula is

$$L_{ave} = \frac{t_1 + t_2 + ... + t_{N_t}}{N_t}. \quad (1)$$

If L_{ave} is less than a threshold latency L_{th}, the edge controller forwards VSD packets and the vehicle controller is switched to the cloud. Otherwise, the edge controller stops forwarding VSD packets and the vehicle controller is switched to the edge server.

However, if the edge controller controls the vehicles, it cannot monitor the latency, because it sends no VSD packet. Therefore, edge and cloud controllers periodically communicate with the send ratio S_r to monitor the latencies.

2.4. Sharing internal states

In this paper, the vehicle controller uses Proportional Integral Differential (PID) control[9]. Therefore, when the vehicle controller is switched, control parameters for PID need to be forwarded as internal states between edge and cloud servers in addition to VSD packets. The control parameters are as follows:

- Last position, velocity and yaw of the vehicles
- Target position using the PID in the next step
- Cumulative position error for PID

2.5. Network architecture

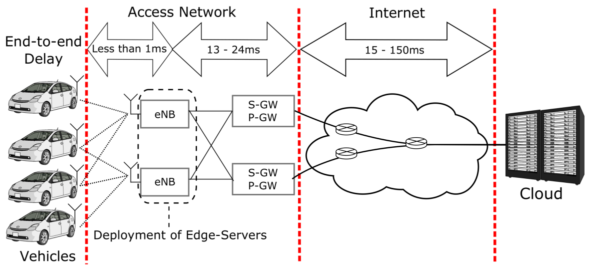


Fig. 3 5G Mobile Network

Our proposed system assumes 5G mobile network (5G) as shown in Fig. 3. Access network is the part of a telecommunications network that connects vehicles to the Internet. There is discussion in 5G mobile development that we can deploy edge servers near eNB. The radio portion of the end-to-end latency is targeted at less than 1ms according the white papers of 5G mobile networks[5, 10].

Next section is described about the prototype of the proposed system.

3. PROTOTYPE USING THE MICRO-CAR

Fig. 4 shows the prototype using a micro-car. The prototype consists of a micro-car platform, a FLARE[11], a WAN emulator that can simulate the Internet environment and a Linux server. FLARE is the programmable network router used as the edge server. The Linux server emulates a cloud server. The proposed system consists of the FLARE and the Linux server.

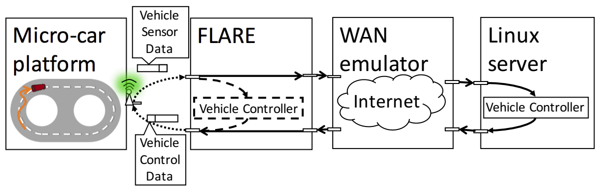


Fig. 4 The prototype using the micro-car

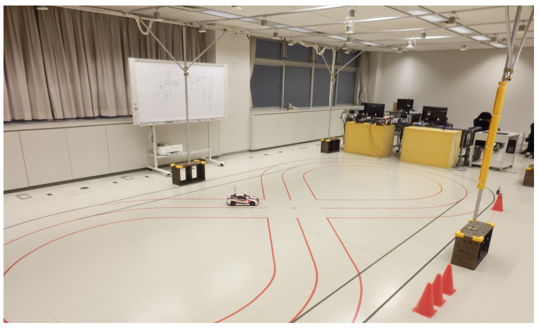


Fig. 5 The micro-car platform

The micro-car platform [12] runs micro-cars (Fig. 5). A micro-car is a radio controlled vehicle that size is 1/10 scale (19cm × 42cm) of the real one. The micro-cars periodically send VSD packets and receives VCD packets to control itself. Position sensing uses ultrasonic with precision errors within 2cm. The ultrasonic sensor is attached to center of the micro-cars. Fig. 6 shows the course and X and Y axis shows space coordinates[m]. The micro-car platform connects FLARE using Wireless LAN as an alternative to 5G radio access and the communication latency and the packet loss ratio are sufficiently small.

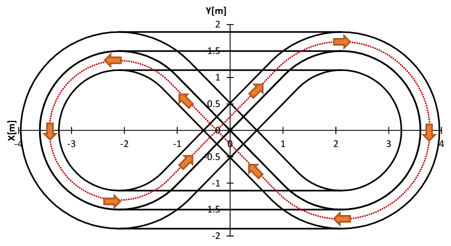


Fig. 6 The course of the micro-car platform

In the prototype, FLARE is used as the edge server. FLARE is the programmable network router and suitable for edge processing[11]. Although usual network switches can only do packet processing, such as packet routing, QoS scheduling and so on, FLARE has the function of not only packet processing but also versatile processing using many core processors. FLARE can execute functions programmed in "Click Modular Router[13] (Click)" that is a programming model for building flexible and configurable routers. In this paper, we implement the vehicle controller in Click.

The Linux server emulates a cloud server, runs CentOS and can also execute Click. FLARE and Linux server can execute equivalent vehicle control interchangeably.

In this prototype, WANem[14] is deployed as a WAN emulator between FLARE and the Linux server to add communication load to VSD and VCD packets.

4. EVALUATION

In this section, we evaluate the proposed system. First, we measure *the stability of driving trajectory* to investigate the effect of the latency. Next, we measure *(i) the stability of driving trajectory, (ii) control ratio* and *(iii) the amount of traffic.*

4.1. Investigation about the effect of the latency

We compare the stability of driving trajectory when applying two control methods, (I) with only the cloud controller and (II) with only the edge controller. A micro-car goes along an 8-shaped course 6 times in each measurement. As Table 1 shows, velocity of the micro-car is 1 m/s , transmission cycle of a VSD packet is 100ms and WANem adds extra latency ranging from 50ms to 300ms.

Table 1 Measurement Parameters

Parameter	Value
Laps	6
Velocity	1 [m/s]
Transmission cycle	100 [ms]
Latency	50, 100, 150, 200, 250, 300 [ms]

4.1.1. How to evaluate the driving trajectory

We use "gap" between driving and desired trajectory to evaluate the stability of the driving trajectory as shown in Fig. 7. The solid line is the driving trajectory, circles represent the positions sent as the VSD packets and the dotted line shows center of the road which are the desired trajectory. The gap is defined as the perpendicular distance between the trajectory and the dotted line. The platform represents one-tenth scale of the real environment including the size of the road width, that of micro-car, and the gap, thus we can easily translate the gap in the experiment into the real gap by multiplying by ten. In addition, the speed of the micro-car, 1 m/s, in the prototype platform corresponds to 36 km/hour in the real environment.

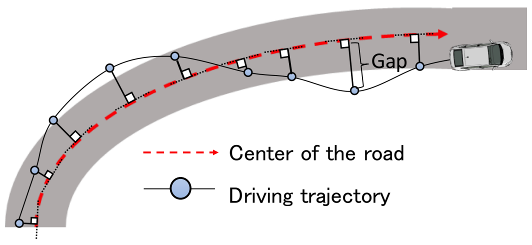


Fig. 7 The definition of the gap: the perpendicular distance between the trajectory and the dotted line

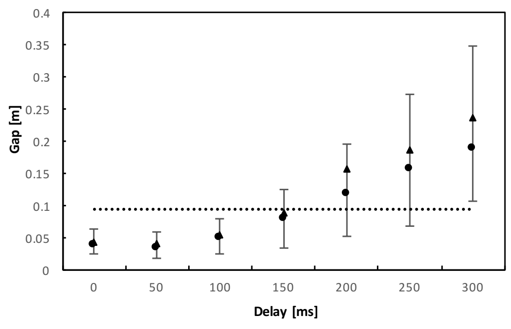


Fig. 8 Latency vs Gap (micro-car scale)

4.1.2. Result

Fig. 8 shows the gap by the cloud as a function of the latency. X-axis is the size of latency[ms] by WANem and Y-axis is the size of gap[m]. The filled circles show the median of the gap, the upper adjacent shows 3/4 quantile and the lower adjacent shows 1/4 quantile. The filled triangles show average. The horizontal dotted line shows the threshold of the gap, 0.095m, that is the limit for keeping the micro-car within the course. The result of the latency 0ms is equivalent to that by the edge server, because the edge server is not effected by latency.

Fig. 8 shows the longer latency is added, the larger the gap becomes. If latency is shorter than 100ms, the micro-car runs without sliding off the course. However, if latency is larger than 150ms, the micro-cars deviate by over 0.095m from the course for the 40% of the entire trajectory.

Fig. 9 shows the driving trajectory. Fig. 9(a), 9(b) and 9(c) show latency is 0ms, 100ms and 150ms, respectively. X and Y axis show space coordinates[m]. In Fig. 9(a), the micro-car runs smoothly along center of the course. However, in Fig. 9(b), the gap increases when the micro-car enters and leaves the curve. Furthermore, in Fig. 9(c), the micro-car slides off the course when the micro-car leaves the left curve (around (-0.5, -0.5)).

From the above, the stability of the driving trajectory controlled by the cloud gets worse as the latency of the Internet get large. However, the edge controller maintains the stability of the driving trajectory, whenever the Internet includes large latency.

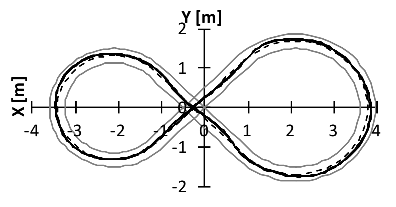
As Fig. 3 shows, the vehicle needs time up to 348ms from sending a VSD packet to receiving a VCD packet through the cloud and only the cloud controller cannot achieve stable control.

4.2. Evaluation about the proposed system

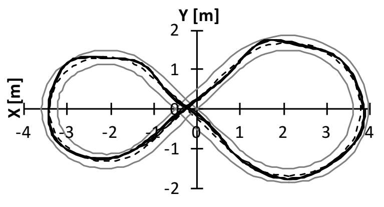
We compare four control methods as shown in Table 2 to show effect of the proposed system. We consider

Table 2 Measurement Methods

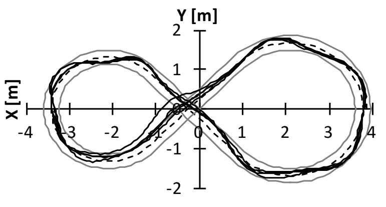
(I)	Vehicle control by using only the cloud controller
(II)	Vehicle control by using only the edge controller
(III)	Automatic switching without sharing internal states
(IV)	Automatic switching with sharing internal states



(a) Edge Server (0ms)



(b) 100ms



(c) 150ms

Fig. 9 Driving Trajectory

three metrics to evaluate the automatic switching: *(i) the stability of driving trajectory, (ii) control ratio* and *(iii) the amount of the traffic*.

The control ratio defines the ratio of the time spent for edge or cloud control to the entire time of control. WANem alternately adds 0ms and 150ms latency every 10 seconds.

Table 3 shows measurement parameters. A micro-car goes along the 8-shaped course 6 times in each measurement. The velocity of the micro-car is 1 [m/s] and transmission cycle of a VSD packet is 100 [ms]. Threshold latency L_{th} of method (III) and (IV) is 100ms. If average latency L_{ave} exceeds 100ms, the edge server controls the vehicle. Otherwise, the cloud controls the vehicle. Window size N_t is 10. Only method (IV) is measured by changing S_r. Send ratio S_r ranges from 0.02 to 1.0.

Table 3 Measurement Parameters

Parameter	Value
Laps	6
Velocity	1[m/s]
Transmission cycle	100[ms]
Threshold latency L_{th}	100[ms]
Window Size N_t	10
Send Ratio S_r	0.02, 0.05, 0.1, 0.2, 0.5, 1.0

4.2.1. Comparison among four control methods

Fig. 10(a) shows the gap of four control methods. X-axis shows control methods and Y-axis shows the gap [m]. S_r and N_t of method (III) and (IV) are 0.2 and 10. In Fig. 10(a), automatic switching with sharing internal achieves equivalent stable control of vehicles with only the edge controller. The average gap of method (II) and (IV) are 0.043m and 0.046m and method (IV) has the stability of the driving trajectory close to method (II). On the other hand, method (III) has larger gap than method (I) in spite of using the edge server. This is caused by not sharing control parameters between edge and cloud servers. In the PID control, the size of cumulative position error affects the magnitude of the control. When the control of the micro-car switches in the sharp curve, the driving trajectory has large gap.

Fig. 10(b) shows the control ratio of four control methods. X-axis shows control methods and Y-axis shows the control ratio. The solid line shows control ratio of the cloud and the dotted line shows control ratio of the edge server. In Fig. 10(b), automatic switching achieves distribution of computational resource usage between edge and cloud servers. Method (I) and (II) consume one or the other of the computational resources. However, control ratio between cloud and edge server using the method (III) and (IV) are 0.43:0.57 and 0.46:0.54. In method (III) and (IV), control ratio of the edge server is larger than that of the cloud. This is caused by S_r. When the edge server controls the micro-car, the edge server forwards VSD packets to the cloud only once in $1/S_r$. Therefore, the update time of the L_{ave} during the edge server control mode varies according to S_r. As a result, control ratio of the edge server is larger than that of the cloud. However, time to update the average latency can be adjusted using send ratio S_r. 4.2.2 describes about it.

Fig. 10(c) shows the traffic of four control methods. X-axis shows control methods and Y-axis shows the amount of the traffic [kbps]. In Fig. 10(c), the amount of the traffic is proportional to control ratio of the cloud. The amount of the traffic of the method (IV) is larger than it of the method (III), because of traffic about sharing control parameter.

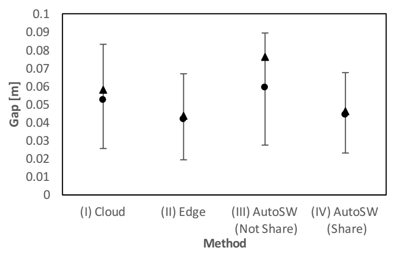
From the above, automatic switching with sharing internal states can achieve equivalent the stability of the driving trajectory by the only edge controller and distribution of computational resource usage between edge and cloud servers.

4.2.2. Effect of send ratio

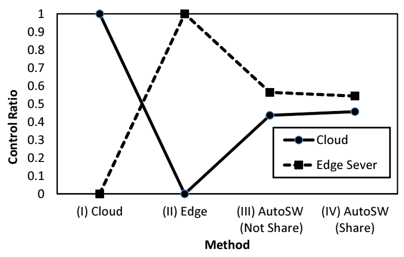
Fig. 11 shows the gap[m], the control ratio and traffic[kbps] as a function of S_r. N_t is 10. In Fig. 11, there is a trade-off between quick update of the average latency and the amount of the traffic. Fig. 11(a) shows the gap is not affected by the change of S_r. However, in Fig. 11(b), S_r affect the control ratio and the amount of the traffic. When the edge server controls the micro-car, the update time of the L_{ave} varies according to S_r. The larger S_r is, the more quickly L_{ave} is updated. When S_r is 1.0, the update time of L_{ave} during the edge server control mode is equivalent with that during the cloud control mode. As a result, the larger S_r is, the larger the control ratio of the cloud becomes.

5. CONCLUSION

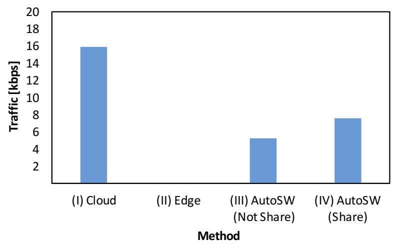
In this paper, we propose “Infrastructure-based vehicle control system” that applies the edge server as one of the controller nodes to cloud-based vehicle control system. The proposed system has an automatic switching method of vehicle controller between edge and cloud



(a) Method vs Gap

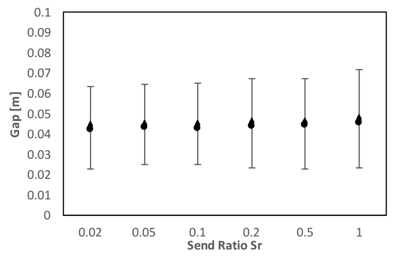


(b) Method vs Control Ratio

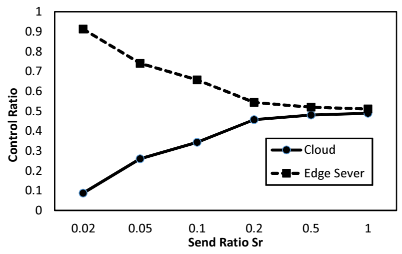


(c) Method vs Traffic

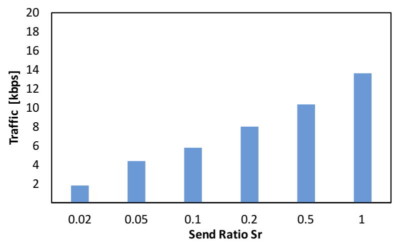
Fig. 10 Comparison among four control Methods



(a) S_r vs Gap



(b) S_r vs Control Ratio



(c) S_r vs Traffic

Fig. 11 Effect of Send Ratio S_r

servers to reduce latency and to balance computational load. We implement the prototype system using micro-cars and evaluate the stability of the driving trajectory, the control ratio, and the amount of the traffic.

As a result, when controlled from the cloud controller with 150ms latency, micro-cars deviate by over 0.095m from the course for the 40% of the entire trajectory, possibly causing car accidents. On the other hand, the edge controller stabilizes the driving trajectory. Also, our proposed system automatically switches control from the cloud and from the edge server according to the network condition without degrading the stability in driving trajectory. Even when the ratio of time of the control by the edge server to that by the cloud server is suppressed to 54%, we can achieve almost the same stability as fully controlled by the edge server.

We plan to consider the following as future work. First, we introduce the other method of averaging latency, such as Exponentially Weighted Moving Average, to smoothly update average latency. Second, we replicate the Internet on the WAN emulator to evaluate the proposed system under realistic condition. At last, we increase in the number of micro-cars to evaluate scalability of the proposed system.

REFERENCES

[1] N. Suzuki et al., “A Low Latency Processing Method on Stream-Based LDM Toward Cloud-Based Automated Driving” *Embedded System Symposium (in Japanese)*, 2015.

[2] M. Patel at el., “Mobile-Edge Computing Introductory Technical White Paper”, *European Telecommunications Standards Institute*, 2014.

[3] E. Guizo, “How google’s self-driving car works”, *IEEE Spectrum Online*, Vol. 18, 2011.

[4] A. Broggi et al., “Autonomous vehicles control in the VisLab intercontinental autonomous challenge”, *Annual Reviews in Control*, Vol. 36, No. 1, pp. 161–171, 2012.

[5] 5GMF, “Networking Technologies for 5G” *The Fifth Generation Mobile Communications Promotion Forum*, 2015.

[6] Y. C. Hu et al., “Mobile Edge Computing A key technology towards 5G”, *European Telecommunications Standards Institute*, 2015.

[7] S. Kumar et al., “Carspeak: a content-centric network for autonomous driving”, *ACM SIGCOMM Computer Communication Review*, Vol. 42, No. 4, pp. 259–270, 2012.

[8] M. Gerla et al., “Internet of Vehicles: From Intelligent Grid to Autonomous Cars and Vehicular Clouds”, *IEEE World Forum on Internet of Things (WF-IoT)*, pp. 241–246, 2014.

[9] M. Araki, “PID control”, *Control systems, robotics ant automation*, Vol. 2, pp. 1–23, 2002.

[10] NTT Docomo, INC., “DOCOMO 5G White Paper”, *NTT DOCOMO, INC.*, 2014.

[11] A. Nakao, “Revisiting “Clean-State” Approach to Re-designing the Internet”, *The 14th GENI Engineering Conference*, 2012.

[12] K. Sasaki et al., “Vehicle Control System based on Mobile-Edge Computing”, *Network System Society Technical Report of IEICE (in Japanese)*, pp. 323–328, 2016.

[13] E. Kohler et al., “The Click Modular Router”, *ACM Transactions on Computer Systems, 2000.*

[14] T. C. Services, “WANem The Wide Area Network emulator”, http://wanem.sourceforge.net .