Dynamic Load Balancing for Energy-Delay Tradeoff in a Cloud-RSU-Vehicle Architecture

Pyeongjun Choi
DGIST
Daegu, South Korea
pyeongjun.choi@dgist.ac.kr

Pildo Yoon
DGIST
Daegu, South Korea
yoonpildo@dgist.ac.kr

Jeongho Kwak
DGIST
Daegu, South Korea
jeongho.kwak@dgist.ac.kr

***Abstract*—Advanced autonomous driving services at Level 4 and above eliminate the need for constant driver supervision, enabling vehicles to respond autonomously to various driving scenarios. The integration of components under edge environment such as the On Board Unit (OBU), Road Side Units (RSUs), and cloud infrastructure, facilitated by V2X communication, enhances perception, decision-making, and control capabilities. However, existing standards lack guidance on efficient computing resource management in autonomous driving systems. To address this, we propose a dynamic computing load balancing algorithm for the cloud-RSU-vehicle architecture. This algorithm optimizes resource allocation and utilization, considering network conditions, computational capabilities, and processing queues using Lyapunov optimization techniques. The stability of the algorithm and trade-off between processing delay and energy consumption is inspected through simulation.**

***Index Terms*—Load balancing, Lyapunov optimization, Edge computing**

I. Introduction

Level 4 and higher autonomous driving services are characterized by advanced automation and the elimination of the need for constant driver supervision. These services aim to enable autonomous vehicles to respond to various driving scenarios and unexpected situations without direct human control. To achieve this, autonomous driving technologies require the integration of multiple components, including the On Board Unit (OBU) of the vehicle, neighboring vehicles, Road Side Units (RSUs), and cloud infrastructure, facilitated by V2X communication. These components collaborate to exchange data, enhance the perception, decision-making, and control capabilities of autonomous vehicles, and ensure effective responses to dynamic road conditions.

The implementation of these technologies involves the utilization of both network resources and computing resources, enabling tasks such as information sharing between adjacent vehicles and processing/transmission of sensor data. However, the existing standards lack comprehensive guidelines on managing computing resources effectively in the context of autonomous driving systems. This oversight poses a significant challenge as the demand for computing power increases with the complexity of autonomous driving operations.

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2023-0-01053, Development of Network Load Balancing Techniques Based on Multiple Communication/Computing/Storage Resources)

To address this challenge, we propose a dynamic computing load balancing algorithm for cloud-RSU-vehicle architecture. The algorithm aims to minimize the energy consumption of vehicle while maintaining finite service time by optimizing the allocation and utilization of computing resources in hierarchical vehicle edge environments. By dynamically distributing computing loads between vehicles, RSU, and cloud, our proposed algorithm can enhance the overall performance, efficiency, and reliability of autonomous driving systems. This adaptive load balancing mechanism takes into account various factors such as network condition, computational capabilities and processing queues of vehicles, RSU, and cloud to make intelligent resource allocation decisions.

II. System Model

A. Task and Arrival Model

We assume a time-slotted system $t = \{0, 1, 2, ..\}$ where the duration of each time slot is Δt. We assume graphic-intensive tasks (i.e. object detection) are generated by vehicles for each time slot t. The size of task arrival $a_i(t)$ is represented in bits, and it is independent and identically distributed for all time slots. The arrival follows $\mathbb{E}[a_i(t)] = \lambda$ and is bounded as $a_i(t) \leq a_{max}$. Each task could be processed by GPU in the vehicle or offloaded to nearby RSU or remote cloud server.

B. Processing and Networking Model

We consider vehicles with GPU which supports DVFS. Each vehicle can adjust its GPU clock frequency $s_i(t) \in \{s_{i,1}, s_{i,2}, ..., s_{max}\}$ (in cycles/Δt) at every time slot t. We assume a single graphic-intensive task, so every arrival has the same processing density γ, which is the number of cycles needed to process a single bit. Note that the processing density does not change among processing units (i.e. GPU in the vehicle, RSU, Cloud). The vehicle can choose only one among three options, local computing, offload to RSU, or offload to Cloud. The scheduling indicator for offload to RSU and Cloud is $\theta_{ij}(t) \in \{0, 1\}, \sigma_i(t) \in \{0, 1\}$. When RSU j is chosen, $\theta_{ij} = 1$ and $\sigma_i = 0$. If the vehicle decides to process within its own GPU, $\theta_{ij} = 0$ and $\sigma_i = 0$. For offloading, we assume OFDMA so that there is no interference among vehicles. Our GPU and network energy models are as follows.

$p_i^u(s_i(t)) = \alpha s_i(t)^3 + \beta$ and $p_i^n(\Sigma_j \theta_{ij}(t) + \sigma_i(t))$. Since the vehicle chooses single offload option, $\Sigma_j \theta_{ij}(t) + \sigma_i(t) \leq 1$ holds for every time slot t. We assume that the vehicle uses constant energy for offloading either to RSU or Cloud and network speed changes by dynamic channel conditions, upper bounded with r_{max}, o_{max}.

C. Queue Model

There exists separate processing queues (in bits) for each component in cloud-RSU-vehicle architecture. The queueing dynamics of our system is as follows.

$$Q_i(t+1) = \Bigg[Q_i(t) + a_i(t) - \frac{s_i(t)(1 - \Sigma_j \theta_{ij}(t) - \sigma_i(t))}{\gamma} - \Sigma_j r_{ij}(t)\theta_{ij}(t) - o_i(t)\sigma_i(t) \Bigg]^+ \tag{1}$$

$$Q_j(t+1) = \left[Q_j(t) + \Sigma_i r_{ij}(t) - \frac{s_j}{\gamma} \right]^+ \tag{2}$$

$$Q_K(t+1) = \left[Q_K(t) + \Sigma_i o_i(t)\sigma_i(t) - \frac{s_K}{\gamma} \right]^+ \tag{3}$$

where Q_i, Q_j, Q_K stands for the processing queue of vehicle, RSU and cloud respectively and $[x]^+ = max(x, 0)$.

III. Dynamic Load Balancing Algorithm

A. Problem Formulation

Our objective is to minimize the energy consumption of vehicle, while maintaining finite service time. To achieve this goal, we control GPU clock frequency and offloading decision according to the network condition and size of the remaining tasks (i.e. queue length). We state our long-term optimization as follows.

$$\textbf{(P1):} \min_{(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{\sigma})} \lim_{T\to\infty} \frac{1}{T} \Sigma_t \Sigma_i \{ p_i^u(s_i(t))(1 - \Sigma_j \theta_{ij}(t) - \sigma_i(t)) + p_i^n(\sigma_i(t) + \Sigma_j \theta_{ij}(t)) \}$$

$$s.t.\ \text{(C1)}: \lim_{T\to\infty} \left\{ \frac{1}{T} \Sigma_{t=0}^{T-1} Q_i(t) + \Sigma_{j=1}^{J} Q_j(t) + Q_K(t) \right\} < \infty$$

$$\text{(C2)}: \sigma_i(t) + \Sigma_j \theta_{ij}(t) \leq 1, \forall i \forall t$$

where $(\boldsymbol{s}, \boldsymbol{\theta}, \boldsymbol{\sigma}) \triangleq (s_i(t), \theta_{ij}(t), \sigma_i(t) : i \in \mathcal{I}, j \in \mathcal{J}, t \in \{0, 1, ...\infty\})$. Constraint (C1) means the queue lengths should be finite (i.e. service time is finite) and Constraint (C2) means only one option could be chosen among 3 offloading options.

B. Algorithm Design

Slot-by-slot objective function We transform original objective function using Lyapunov optimization techniques. First, we define Lyapunov function as follows.

$$L\big(\boldsymbol{Q}(t)\big) = \frac{1}{2}\Sigma_{i=1}^{I} Q_i(t)^2 + \frac{1}{2}\Sigma_{j=1}^{J} Q_j(t)^2 + \frac{1}{2} Q_K(t)^2 \tag{4}$$

where $\boldsymbol{Q}(t) = \{Q_i(t), Q_j(t), Q_K(t)\}$ Then we define Lyapunov drift using (4) as follows.

$$\Delta\Big(L\big(\boldsymbol{Q}(t)\big)\Big) = \mathbb{E}\Big\{ L\Big(\boldsymbol{Q}(t+1)\Big) - L\Big(\boldsymbol{Q}(t)\Big) | \boldsymbol{Q}(t) \Big\} \tag{5}$$

To reflect the penalty (i.e. energy consumption of vehicle) to our objective function, we define Lyapunov drift-plus-penalty function as follows.

$$\Delta\Big(L\big(\boldsymbol{Q}(t)\big)\Big) + V\mathbb{E}\Big\{ \Sigma_i p_i^u \Big(s_i(t)\Big(1 - \Sigma_j \theta_{ij}(t) - \sigma_i(t)\Big)\Big) + p_i^n\big(\sigma_i(t) + \Sigma_j \theta_{ij}(t)\big) \Big\} \tag{6}$$

where V is a trade-off parameter between the energy consumption of the vehicle and queue stability. Then we can optimize the energy consumption and queue stability by minimizing single objective function (6).

Upper bound of objective function We derive the upper bound of the objective function (6) with the queue model (1 $\sim$ 3) and arrival model.

Lemma 1: under any possible control variables $(\theta_{ij}(t), o_i(t), s_i(t))$, we have:

$$\begin{aligned}
&\Delta\Big(L\big(\boldsymbol{Q}(t)\big)\Big) + V\mathbb{E}\Big\{ \Sigma_i p_i^u \Big(s_i(t)\Big(1 - \Sigma_j \theta_{ij}(t) - \sigma_i(t)\Big)\Big) + p_i^n\big(\sigma_i(t) + \Sigma_j \theta_{ij}(t)\big) \Big\} \\
&\leq B + V\mathbb{E}\Big\{ \Sigma_i p_i^u \Big(s_i(t)\big(1 - \Sigma_j \theta_{ij}(t) - \sigma_i(t)\big)\Big) + p_i^n\big(\sigma_i(t) + \Sigma_j \theta_{ij}(t)\big) | \boldsymbol{Q}(t) \Big\} \\
&- \mathbb{E}\Big\{ \Sigma_i \Big(\frac{s_i(t)\big(1 - \Sigma_j \theta_{ij}(t) - \sigma_i(t)\big)}{\gamma} + \Sigma_j r_{ij}(t)\theta_{ij}(t) + o_i(t)\sigma_i(t) - a_i(t) \Big) Q_i(t) | \boldsymbol{Q}(t) \Big\} \\
&- \mathbb{E}\Big\{ \Sigma_j \Big(\frac{s_j}{\gamma} - \Sigma_i r_{ij}(t)\theta_{ij}(t) \Big) Q_j(t) | \boldsymbol{Q}(t) \Big\} \\
&- \mathbb{E}\Big\{ \Big(\frac{s_K}{\gamma} - \Sigma_i o_i(t) \Big) Q_K(t) | \boldsymbol{Q}(t) \Big\}
\end{aligned} \tag{7}$$

$$\text{where } B = \frac{1}{2}\Bigg(I\Big(a_{max}^2 + \frac{s_{max}^2}{\gamma^2} + r_{max}^2 + o_{max}^2 \Big) + J\Big(\frac{s_j^2}{\gamma^2} + r_{max}^2 \Big) + o_{max}^2 + \frac{s_K^2}{\gamma^2} \Bigg).$$

Proof: The proof can be done similarly as in [1]. ■

IV. Dynamic Load Balancing Algorithm

We derive the algorithm that minimizes the upper bound (i.e. right-hand side of (7)). We describe the mechanism of the algorithm with pseudocode in algorithm 1.

At each time slot t, for each vehicle i, the dynamic load balancing algorithm calculates the objective function (6) for

Algorithm 1 Dynamic Load Balancing

At each time slot t, for all vehicle i,
calculate
$A = \min_{s(t)} Vp_i^u(s_i(t)) - \left(\frac{s_i(t)}{\gamma} - a_i(t)\right)Q_i(t)$
$B = Vp_i^n(1) - \big(r_i(t) - a_i(t)\big)Q_i(t) + o_i(t)Q_j(t)$
$C = Vp_i^n(1) - \big(o_i(t) - a_i(t)\big)Q_i(t) + o_i(t)Q_K(t)$
if $A = min(A, B, C)$ **then**
  Do local computing $(\theta_{ij}(t) = 0, o_i(t) = 0)$
else if $B = min(A, B, C)$ **then**
  Offload to RSU $(\theta_{ij}(t) = 1, o_i(t) = 0)$
else if $C = min(A, B, C)$ **then**
  Offload to Cloud $(\theta_{ij}(t) = 0, o_i(t) = 1)$
end if

possible cases (i.e. local computing, offload to RSU, offload to cloud). Note that for local computing, we can easily find GPU clock frequency that minimizes the objective function since the objective function is convex in our area of interest.

V. Evaluation

A. Simulation Setup

We consider a single cloud server, 2 RSUs, and 4 vehicles. Each vehicle generates task following normal distribution with mean 3.12 Mbit and standard deviation 0.5 Mbit. Network speed $r_i(t)$ and $o_i(t)$ follows normal distribution with mean 2.4 Mbit and standard deviation 0.48 Mbit. Each vehicle can choose GPU clock frequency with the range of [100 MHz, 1000 MHz]. The processing density of the task γ is set as 167. We compare proposed algorithm with **(1) Only offloading**: choose better offloading target among RSU and cloud based on (6) and **(2) Only local**: local computing with GPU clock frequency chosen by (6).

B. Simulation Result

Fig. 1 shows the characteristics of our algorithm. Over time, we can see that the queue length and GPU clock frequency of each vehicle converge to the equilibrium point. In addition, the energy consumption of each vehicle in fig. 1 shows that the proposed algorithm achieves fairness by efficiently distributing the load for each vehicle according to the network speed and queue length. Fig. 2 shows the impact of trade-off parameter V in our algorithm and the difference with the comparison algorithm (i.e. only local). Only offloading algorithm is not included in the graph because it fails to achieve queue stability. By adjusting V, we can utilize the trade-off between queue length and energy consumption. We can see that our dynamic load balancing algorithm achieves 13% lower processing queue with similar energy consumption.

VI. Conclusion

In this paper, we proposed dynamic load balancing algorithm for cloud-RSU-vehicle architecture using Lyapunov optimization and showed operating characteristics of algorithm

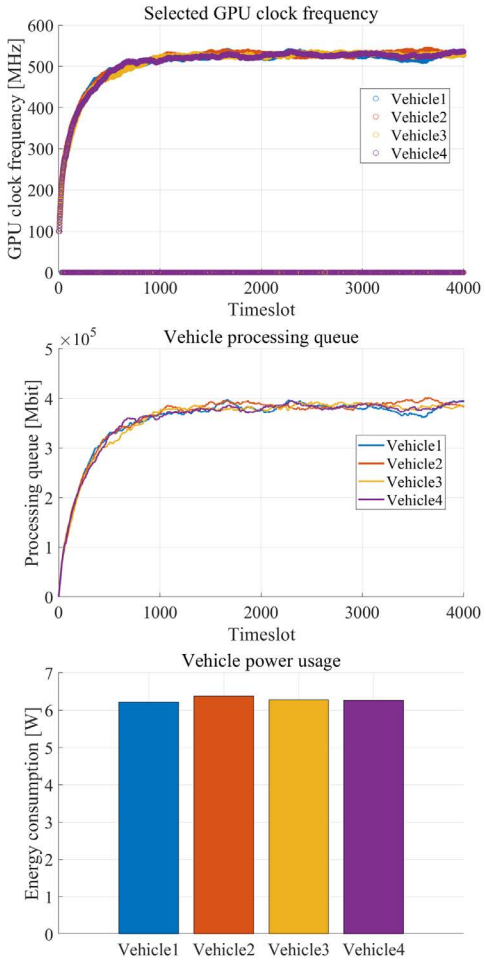

Fig. 1. Operating characteristics of dynamic load balancing algorithm

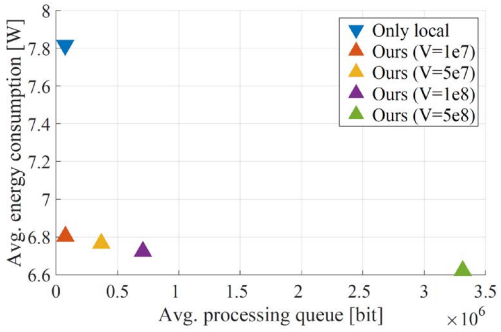

Fig. 2. Trade-off between processing queue and energy consumption

and trade-off between processing queue and energy consumption through simulation. We expect that this technology will play a crucial role in advancing the capabilities of autonomous driving systems, improving safety, and providing a reliable autonomous driving experience for users.

References

[1] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.