https://doi.org/10.7840/kics.2024.49.3.377

클라우드-RSU-차량 아키텍처에서 에너지 지연 균형을 위한 동적 로드밸런싱 알고리즘

최 평 준◆, 윤 필 도*, 곽 정 호°

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

Pyeongjun Choi◆, Pildo Yoon*, Jeongho Kwak°

요 약

레벨 4 이상의 첨단 자율주행 서비스는 운전자의 지속적인 개입을 필요로 하지 않으며 차량이 자율적으로 다양한 주행 시나리오에 대응할 수 있도록 한다. V2X 통신을 활용한 On Board Unit (OBU), Road Side Units (RSU) 및 클라우드 인프라와 같은 Vehicle Edge Computing (VEC) 구성 요소의 통합을 통해 각 구성 요소에 산재된 네트워킹 자원과 컴퓨팅 자원을 효율적으로 사용하여 차량의 사물 인식, 의사 결정 및 제어 기능의 성능을 향상시킬 수 있다. 하지만 기존 LTE-V2X나 5G NR V2X 표준은 자율 주행 시스템의 네트워킹 자원에 관해서만 집중하여 다루고 있으며 차량 혹은 RSU의 연산 능력이나 태스크의 처리 밀도 등 컴퓨팅 자원에 관해서는 거의 다루지 않는다. 이를 해결하기 위해 우리는 클라우드-RSU-차량 아키텍처를 위한 동적 컴퓨팅 로드밸런싱 알고리즘을 제안한다. 제안 알고리즘은 Lyapunov 최적화 기술을 사용하여 네트워크 상태, 차량의 연산 능력 및 처리 대기열을 함께 고려하여 자원 할당과 오프로딩을 동적으로 결정함으로써 평균 처리 지연을 일정 수준으로 유지하는 동시에 차량의 평균 에너지 소모를 최소화한다. 자율주행 시나리오에서의 시뮬레이션을 통해 알고리즘이 동적으로 변화하는 네트워크 속도와 서비스 요청에 적응적으로 동작하여 차량 내에서 처리하는 방법에 비해 처리 지연을 비슷하게 유지하면서 차량의 에너지 소모를 훨씬 더 많이 줄일 수 있음을 보인다.

키워드 : 로드밸런싱, Lyapunov 최적화, 엣지 컴퓨팅
Key Words : Load balancing, Lyapunov optimization, Edge computing

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 by efficiently utilizing networking and computing resources scattered across each component. However, existing standards like LTE-V2X or 5G NR V2X focus on networking resources and lack guidance

※ 이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아수행된 연구임 (No.2022-0-01053, 다중 통신기술 네트워크 로드밸런싱 기술개발)
◆ First Author : Daegu Gyeongbuk Institute of Science and Technology, pyeongjun.choi@dgist.ac.kr, 학생회원
° Corresponding Author : Daegu Gyeongbuk Institute of Science and Technology, jeongho.kwak@dgist.ac.kr, 종신회원
* Daegu Gyeongbuk Institute of Science and Technology, yoonpildo@dgist.ac.kr, 학생회원
논문번호 : 202310-114-C-RU, Received October 25, 2023; Revised November 19, 2023; Accepted November 27, 2023

on efficient computing resource management in autonomous driving systems. To tackle this, we propose a dynamic computing load balancing algorithm for the cloud-RSU-vehicle architecture. The proposed algorithm minimizes the average energy consumption of the vehicle while keeping the average processing delay under a certain level through resource allocation and offloading decisions considering network conditions, computational capabilities, and processing queues using Lyapunov optimization techniques. Via simulations under autonomous vehicle scenario, we show that the proposed algorithm operates adaptively to dynamically changing network speeds and service requests; thereby it significantly saves the average energy consumption with similar average processing delay compared to a policy of only vehicle computing.

Ⅰ. 서 론

레벨 4 이상의 자율주행 서비스는 자동화가 고도화되어 운전자의 지속적인 개입이 필요하지 않다. 자율주행 서비스의 목표는 자율주행차가 인간의 직접적인 통제 없이 다양한 주행 시나리오와 예상치 못한 상황에 대응할 수 있도록 하는 것이다. 이를 달성하기 위해 자율주행 기술에는 V2X 통신을 통한 차량의 OBU(On Board Unit), 주변 차량, RSU(Road Side Unit), 클라우드 인프라를 포함한 여러 구성 요소의 협력이 필요하다. 구성 요소들이 서로 협력함으로써 데이터를 교환하고[1] 효율적으로 자원을 활용하여 자율주행 차량의 인식, 의사 결정 및 제어 기능을 향상시키며[2], 역동적인 도로 상황에 대한 효과적인 대응을 보장할 수 있게 된다. 구체적으로는 인접 차량 간의 정보 공유, 센서 데이터 처리/전송과 같은 작업이 고려될 수 있으며, 이 과정에서 각 구성 요소에 산재된 네트워킹 자원과 컴퓨팅을 효율적으로 활용할 수 있게 된다. 그러나 LTE-V2X (Release 14)나 5G NR V2X (Release 16)과 같은 기존 표준에는 자율 주행 시스템의 네트워킹 자원 관리를 위주로 다루고 있고, 처리 지연이나 정확도 등의 서비스 요구사항을 만족시키기 위한 컴퓨팅 자원 관리는 거의 다루지 않고 있다[3,4]. 미래에는 Li- DAR와 같은 다양한 고성능 센서들이 더 널리 사용 되고 자율 주행 태스크들이 점점 고도화됨에 따라 고성능 컴퓨팅에 대한 수요가 증가할 것으로 예상 되기 때문에 차량이나 RSU의 컴퓨팅 연산 능력과 다양한 서비스 요구사항을 고려한 컴퓨팅 자원 관리는 필수적이다[5].

이러한 문제를 해결하기 위해 우리는 클라우드-RSU-차량 아키텍처를 위한 동적 컴퓨팅 로드밸런싱 알고리즘을 제안한다. 우리는 계층적 차량 엣지 컴퓨팅 (Vehicle Edge Computing, VEC) 환경에서 컴퓨팅 리소스의 할당 및 활용을 최적화하여 유한한 평균 서비스 지연 시간을 보장하면서 차량의 평균 에너지 소비량을 최소화하는 것을 목표로 한다. 제안된 알고리즘은 차량, RSU 및 클라우드 간에 컴퓨팅 부하를 동적으로 분산함으로써 자율주행 시스템의 전반적인 성능, 효율성 및 신뢰성을 향상시킬 수 있다. 구체적으로, 제안된 알고리즘은 네트워크 상태, 컴퓨팅 연산 성능, 차량과 RSU, 클라우드의 처리 대기열을 통합적으로 고려하여 평균 서비스 지연을 일정 수준으로 유지하면서 차량의 평균 에너지 소모를 최소화하는 자원 할당 결정 및 오프로딩 결정을 내린다.

Ⅱ. 연구 배경

최근 차량은 LiDAR, 카메라 등의 고용량 센서 데이터를 처리하기 위해 높은 연산 부하를 겪고 있다. 차량의 연산 부하가 과다한 경우 높은 에너지 소모와 더불어 자율주행 차량의 인식, 의사 결정과 같은 기능의 처리 시간이 증가하며 극단적인 경우 처리가 불가능해진다. 차량의 연산 부하를 완화하기 위한 기술로 차량의 연산 부하를 V2X 통신을 통해 RSU 혹은 클라우드와 같은 서버로 분산 (즉, 오프로딩) 하는 차량 엣지 컴퓨팅이 있다. 하지만, 차량의 모든 연산을 항상 오프로딩할 경우 무선 통신 채널 환경이나 서버의 대기열 길이 등에 따라 단대단 지연이 변화하여 일정한 수준의 서비스 지연을 유지하기 어렵다. 이에 연산 부하를 적절하게 나눠 차량의 에너지 소모 최소화, 단대단 지연 감소를 목표로 하는 로드밸런싱 관련 연구가 활발히 진행되고 있다.

예를 들어, Kwak *et al.*은 Lyapunov 최적화 기법을 사용하여 단말의 프로세서와 통신에 사용되는 에너지를 최소화하면서 다양한 어플리케이션의 대기열 의 안정성을 보장하는 연구를 진행했다[6]. Kumar *et al.*은 Lyapunov 최적화 기법에 기반한 강화학습을 통해 차량과 엣지의 에너지 소비를 최소화하며 대기열의 안정성을 보장하는 알고리즘을 제안했다[7]. Zhang *et al.*은 차량이 오프로딩시 발생하는 통신 요금이나 전력 소모 등의 비용을 최소화하는 오프로딩 정책을 제안하였다[8].

기존 연구들은 RSU-단말 (즉, RSU-차량)로 이루어

진 작은 아키텍처에서 연구를 진행하였다. 또한 여러 단말 간의 자원 소모량 관점에서의 공정성에 대해 다루지 않았으며, 대기열의 안정성을 보장하지 않은 연구도 있었다. 우리는 클라우드-RSU-차량으로 확장된 아키텍처에서 각 구성요소 (즉, 클라우드, RSU, 차량)의 대기열의 안정성을 보장하며 차량의 에너지 소모를 최소화하는 알고리즘을 제안한다. 제안 알고리즘은 또한 각 차량의 부하를 효율적으로 분산시켜 에너지 소모 관점에서 차량 간의 공정성을 달성할 수 있다.

Ⅲ. 시스템 모델

우리는 Fig. 1 와 같은 클라우드-RSU-차량 아키텍처를 가정한다. 각 차량은 RSU, 클라우드와 무선으로 연결되어 있다. 차량은 자신의 태스크를 차량 내부의 GPU, RSU, 그리고 클라우드 중 하나를 골라 처리하도록 할 수 있다. 하나의 클라우드에 다수의 RSU와 차량이 연결되어 있고, 각 RSU도 다수의 차량과 연결되어 있다. 차량은 가장 가까운 RSU와 연결되며, 차량이 이동함에 따라 연결된 RSU는 바뀔 수 있다.

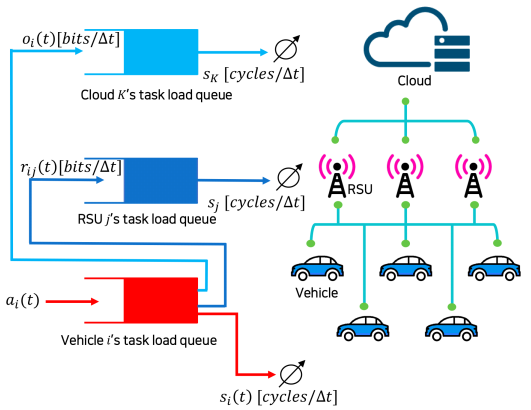

Fig. 1. Cloud-RSU-Vehicle architecture

3.1 태스크 모델.

각 시간 슬롯의 지속 시간이 Δt 인 시간 슬롯 시스템 $t = \{0, 1, 2, ..\}$을 가정한다. 우리는 그래픽 집약적인 태스크 (예: 물체 감지)이 각 시간 슬롯 t에 대해 차량에 의해 생성된다고 가정한다. 태스크의 로드 $a_i(t)$는 비트 단위로 표시되며 모든 시간 슬롯에 대해 독립적이고 동일하게 분포된다. 태스크의 로드는 $\mathbb{E}[a_i(t)] = \lambda$를 따르며 $a_i(t) \le a_{max}$로 제한된다. 각 태스크는 차량의 GPU로 처리되거나 근처 RSU 또는 원격 클라우드 서버로 오프로드될 수 있다.

3.2 프로세싱 및 네트워킹 모델.

우리는 DVFS를 지원하는 GPU가 장착된 차량을 고려한다. 각 차량은 모든 시간 슬롯 t에서 GPU 클럭 주파수 $s_i(t) \in \{s_{i,1}, s_{i,2}, ..., s_{max}\}$ (cycles/Δt 단위)를 조정할 수 있다. 우리는 하나의 그래픽 집약적 태스크를 가정하므로 모든 로드는 단일 비트를 처리하는 데 필요한 사이클 수인 γ와 동일한 처리 밀도를 갖는다. 처리 밀도는 처리 장치(예: 차량의 GPU, RSU, 클라우드) 간에 변하지 않는다. 차량은 로컬 컴퓨팅, RSU로 오프로드, 클라우드로 오프로드의 세 가지 옵션 중 하나만 선택할 수 있다. RSU 및 클라우드로의 오프로드에 대한 결정 변수는 $\theta_{ij}(t) \in \{0, 1\}$, $\sigma_i(t) \in \{0, 1\}$이다. 예를 들어 RSU j를 선택한 경우 $\theta_{ij} = 1$ 및 $\sigma_i = 0$ 이 되며, 혹은 차량이 자체 GPU를 활용하여 GPU 클럭 주파수 $s_i(t)$로 처리하기로 결정한 경 우 $\theta_{ij} = 0$ 및 $\sigma_i = 0$가 된다. 오프로딩을 위해서는 차량 간 간섭이 없도록 OFDMA를 가정한다. 우리 는 GPU 및 네트워크 에너지 모델을 다음과 같이 정의한다. $p_i^u(s_i(t)) = \alpha s_i(t)^3 + \beta$, $p_i^n(\Sigma_j \theta_{ij}(t) + \sigma_i(t))$ 차량이 RSU 혹은 클라우드 중 하나의 오프로딩 옵션을 선택하므로 $\Sigma_j \theta_{ij}(t) + \sigma_i(t) \le 1$은 모든 시간 슬롯 t에 대해 유지된다. 우리는 차량이 RSU 또는 클라우드로의 오프로딩을 위해 일정한 에너지를 사용 하고 r_{max}, o_{max}로 상한되는 동적 채널 조건에 따라 네트워크 속도가 변경된다고 가정한다.

3.3 대기열 모델.

클라우드-RSU-차량 아키텍처의 각 구성 요소에 대해 별도의 처리 대기열(비트 단위)이 존재한다. 우리 시스템의 대기열 역학은 다음과 같다.

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

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

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

여기서 Q_i, Q_j, Q_K 는 각각 차량, RSU 및 클라우드의 처리 대기열을 나타내고 $[x]^+ = max(x, 0)$ 를 나타낸다.

Ⅳ. 동적 로드밸런싱 알고리즘

4.1 문제 정의.

우리의 목표는 시간 평균 서비스 지연을 일정 수준 아래로 유지하면서 차량의 시간 평균 에너지 소비량을 최소화하는 것이다. 이 목표를 달성하기 위해 우리는 네트워크 상태와 남은 로드의 크기(예: 대기열 길이)에 따라 GPU 클럭 주파수와 오프로딩 결정을 제어한다. 우리는 장기적인 관점에서의 최적화 문제를 다음과 같이 정의한다.

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

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

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

여기서 $(\boldsymbol{s}, \theta, \sigma) \triangleq (s_i(t), \theta_{ij}(t), \sigma_i(t) : i\in\mathcal{I}, j\in\mathcal{J}, t\in\{0, 1, ...\infty\})$를 의미한다. 제약 조건 (C1)은 대기열 길이가 유한해야 함을 의미한다. Little's theorem을 활용하면 평균 대기열 길이는 간접적으로 평균 서비스 지연시간으로 해석될 수 있기 때문에 제약 조건 (C1)은 곧 서비스 시간이 유한해야 함을 의미한다[9]. 제약 조건 (C2)는 3가지 오프로딩 옵션 중 하나만 선택할 수 있음을 의미한다.

4.2 알고리즘 도출.

최적화 문제 (P1)은 장기적인 관점에서의 최적화를 다룬 것으로, 임의의 시간 슬롯 t에서 최적의 결정 변수를 얻기 위해서는 미래의 정보까지 필요로 한다. 미래의 정보를 아는 것은 현실적으로 불가능하며, 태스크의 로드나 네트워크 속도 등 변수의 분포나 평균값 또한 알 수 없다. 이에 우리는 변수들의 분포나 미래의 정보 없이 현재의 정보만을 활용하여 점근적으로 최적의 선택을 하도록 동작하는 Lyapunov 최적화 기법[10]을 사용하여 (P1)을 매 시간 슬롯마다 풀 수 있는 문제로 변환, 새로운 목적 함수를 얻는다. 먼저 Lyapunov 함수를 다음과 같이 정의한다.

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

여기서 $\boldsymbol{Q}(t) = \{Q_i(t), Q_j(t), Q_K(t)\}$이다. 이를 사용하여 Lyapunov drift를 다음과 같이 정의한다.

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

페널티(즉, 차량의 에너지 소비량)를 목적 함수에 반영하기 위해 Lyapunov drift plus penalty 함수를 다음과 같이 정의한다.

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

여기서 V는 차량의 에너지 소비량과 대기열 안정성 간의 균형을 결정하는 매개변수이다. 그런 다음 단일 목적 함수 (6)을 최소화하여 에너지 소비량과 대기열 안정성을 최적화할 수 있다. 우리는 목적 함수의 상한을 최소화하여 우리의 목표를 달성한다. 대기열 모델(1 ~ 3)과 로드 모델을 사용하여 목적 함수 (6)의 상한을 다음과 같이 도출할 수 있다.

Lemma 1: 가능한 제어 변수 $(\theta_{ij}(t), o_i(t), s_i(t))$에서 다음을 얻는다.

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

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

Proof: 위의 Lemma는 [6] 논문과 유사하게 유도할 수 있다.

4.3 동적 로드밸런싱 알고리즘.

우리는 목적 함수의 상한(즉, (7)의 오른쪽)을 최소화하는 알고리즘을 다음과 같이 전개한다. 알고리즘 1의 의사 코드를 사용하여 알고리즘의 메커니즘을 설명한다.

Algorithm 1 동적 로드밸런싱 알고리즘

At each time slot t, for all vehicle i,

calculate

$$A = \min_{s(t)} V p_i^u(s_i(t)) - \left(\frac{s_i(t)}{\gamma} - a_i(t)\right) Q_i(t)$$

$$B = \min_{j} V p_i^n(1) - \left(r_{ij}(t) - a_i(t)\right) Q_i(t) + r_{ij}(t) Q_j(t)$$

$$C = V p_i^n(1) - \left(o_i(t) - a_i(t)\right) Q_i(t) + o_i(t) Q_K(t)$$

if $A = min(A,B,C)$ **then**

Do local computing $(\theta_{ij}(t) = 0, \sigma_i(t) = 0)$

else if $B = min(A,B,C)$ **then**

Offload to RSU $(\theta_{ij}(t) = 1, \sigma_i(t) = 0)$

else if $C = min(A,B,C)$ **then**

Offload to Cloud $(\theta_{ij}(t) = 0, \sigma_i(t) = 1)$

end if

각 시간 슬롯 t에서 각 차량 i 에 대해 동적 부하 분산 알고리즘은 각 선택지 (즉, 로컬 컴퓨팅, RSU로 오프로드, 클라우드로 오프로드)에 대한 목적 함수 (6)을 계산한다. 로컬 컴퓨팅의 경우 목적 함수가 관심 영역에서 볼록하므로 목적 함수를 최소화하는 GPU 클럭 주파수를 쉽게 찾을 수 있다. 각 경우의 목적 함수의 값을 비교하여 가장 낮은 값을 달성하는 오프로딩 결정과 GPU 클럭 주파수를 선택한다.

V. 시뮬레이션

5.1 시뮬레이션 환경.

우리는 단일 클라우드 서버, 2개의 RSU 및 4대의 차량을 고려하며, 각 시간 슬롯의 길이는 1초로 가정한다. 각 차량은 평균 2880 Mbits, 표준편차 500Mbits의 정규 분포를 따르는 태스크 로드를 생성한다. 각 태스크의 처리 밀도 γ는 0.02로 설정한다. RSU로 오프로딩할 때의 네트워크 속도 $r_i(t)$는 평균이 2400 Mbits이고 표준 편차가 480 Mbits인 정규분포를, 클로우드로 오프로딩 할 때의 네트워크 속도 $o_i(t)$는 평균이 2040 Mbits이고 표준 편차가 408 Mbits인 정규 분포를 따른다. 각 차량은 [100 MHz, 700 MHz] 범위의 GPU 클럭 주파수를 선택할 수 있다. 각 RSU는 고정된 GPU 클럭 주파수 864 MHz를 사용하여 태스크를 처리하며, 클라우드는 고정된 GPU 클럭 주파수 1440 MHz로 태스크를 처리한다. 우리는 제안 알고리즘을 다음과 비교한다. **(1) Fixed frequency**: 각 차량의 GPU 클럭 주파수를 700 MHz 로 고정하고 식 (6)에 기반하여 로컬 컴퓨팅, RSU, 그리고 클라우드 중 선택한다. **(2) Max throughput**: 각 차량의 GPU 클럭 주파수를 700 MHz로 고정하고 로컬 컴퓨팅, RSU, 그리고 클라우드 중 가장 높은 처리량을 달성하는 것을 선택한다. **(3) Only offloading**: 식 (6)에 기반하여 RSU와 클라우드 중에서 더 나은 오프로드 대상을 선택한다. **(4) Only local**: 식 (6)을 통해 결정된 GPU 클럭 주파수를 사용한 로컬 컴퓨팅만을 선택한다. 우리는 4000 시간 슬롯 동안의 시뮬레이션을 통해 제안한 알고리즘의 시간에 따른 대기열 수렴 여부와 에너지 소비량 등을 보인다.

5.2 시뮬레이션 결과.

Fig. 2 - Fig. 4 는 제안 알고리즘의 특징을 보여준다. Fig. 2를 보면 시간이 지남에 따라 각 차량의 대기열 길이가 수렴하는 모습을 확인할 수 있다. Fig. 3에서는 각 차량의 GPU 클럭 주파수가 수렴하는 것을 확인할 수 있다. 또한 차량 1에 대해 figure를 확대해보면 차량

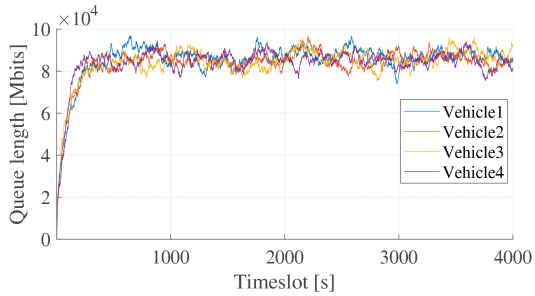

Fig. 2. Processing queue of each vehicle

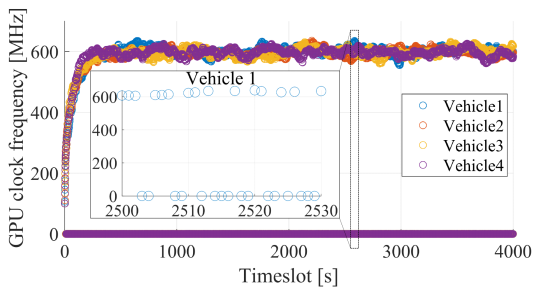

Fig. 3. GPU clock frequency of each vehicle

이 로컬 컴퓨팅과 오프로딩을 적절히 번갈아 사용하는 것을 확인할 수 있다. 이는 제안하는 알고리즘이 동적으로 변화하는 네트워크 속도와 태스크의 로드양, 대기열의 길이에 따라 적응적으로 로컬 컴퓨팅과 RSU 오프로딩, 혹은 클라우드 오프로딩을 선택하여 목표한 대기열 길이를 유지하면서 에너지 소모를 최소화함을 보여준다. 또한, Fig. 4의 각 차량별 에너지 소비량은 제안된 알고리즘이 네트워크 속도와 대기열 길이에 따라 각 차량의 부하를 효율적으로 분산시켜 에너지 소모 관점에서 각 차량 간의 공정성을 달성함을 보여준다.

Fig. 5는 제안 알고리즘의 평균 에너지 소비량와 평균 대기열 길이 간 균형을 위한 매개변수 V의 영향과 비교 알고리즘(즉, Only local, Fixed frequency, Max throughput)과의 차이점을 보여준다. Only offloading의 경우 대기열 안정성을 달성하지 못하고 대기열이 발산하기 때문에 그래프에서 제외하였다. 그래프에서 보이는 것처럼 제안 알고리즘은 V를 조 정함으로써 대기열 길이와 에너지 소비량 간의 균 형점을 조절하여 사용할 수 있으며 비교 알고리즘 대비 적은 에너지 소모로도 비슷한 대기열 길이 (즉, 서비스 지연 시간)을 달성할 수 있다. 구체적으로, Max thorughput 알고리즘과 비슷한 수준의 대기열 길이를 유지하면서 54% 낮은 에너지 소모를 달성 하였다. 또한 Fixed frequency 알고리즘과는 비슷한 수준의 에너지를 소모하면서 44% 낮은 대기열 길이를 달성하였다. Only local 대비 비슷한 에너지 소비량으로 13% 더 낮은 대기열 길이를 달성하는 것을 확인하였다. 이 결과는 제안 알고리즘이 동적으로 변화하는 네트워크 속도와 태스크의 로드양, 대기열의 길이를 함께 고려하여 과도한 에너지 소모를 방지하며 대기열 길이를 적정 수준으로 유지함을 보여준다.

Fig. 6 제안 알고리즘의 차량 대수에 따른 차량의 평균 에너지 소비량과 오프로딩 선택률, 평균 대기열 길이의 변화를 보여준다. 차량 대수를 4대에서 8대, 12대로 늘려감에 따라 평균 대기열 길이는 5.17% 감소하고 오프로딩 선택률은 40.56% 감소하는 한편 에너지 사용량은 19.77% 증가하는 것을 확인할 수 있다. 이는 차량의 수가 늘어나면서 RSU와 클라우드로 오프로딩 되는 태스크의 로드양이 증가하여 RSU와 클라우드의 대기열 길이가 증가했기 때문이다. 즉, 오프로딩 옵션의 서비스 지연 시간이 길어지자 각 차량이 차량 내부에서 에너지를 조금 더 써서 대기열 길이를 줄이려는 시도를 하고 있다는 것으로 해석할 수 있다. 이 결과는 우리의 제안 알고리즘이 차량의 내부 환경 뿐만 아니라 네트워크 속도나 엣지 서버에서의 처리 지연과 같은 외부환경의 변화에도 유연하게 동작할 수 있음을 시사한다.

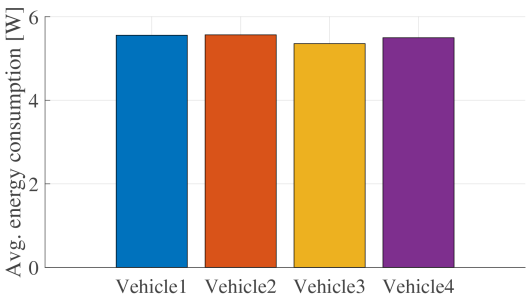

Fig. 4. Average energy consumption of each vehicle

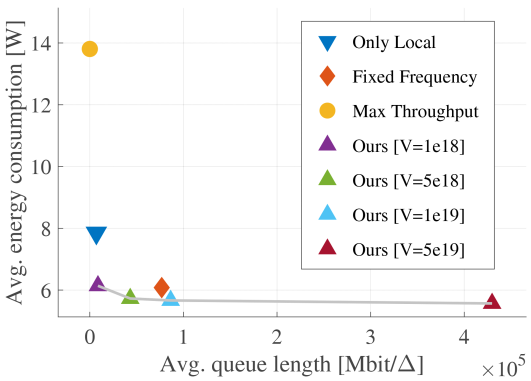

Fig. 5. Trade-off between processing queue and energy consumption

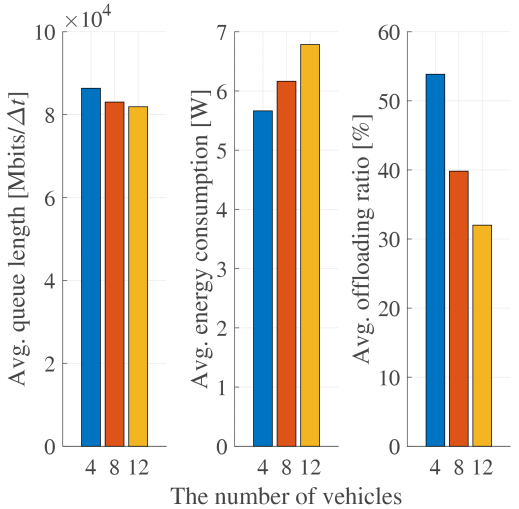

Fig. 6. Average queue length, energy consumption, offloading ratio by number of cars

Ⅵ. 결 론

본 논문에서는 Lyapunov 최적화를 활용한 클라우드-RSU-차량 아키텍처를 위한 동적 로드밸런싱 알고리즘을 제안하고 시뮬레이션을 통해 제안 알고리즘이 동적으로 변화하는 환경 속에서도 오직 차량 내에서 컴퓨팅하는 방안에 비해 비슷한 수준의 평균 서비스 지연을 유지하면서 평균 에너지를 훨씬 더 많이 절약하는 것을 보였다. 우리는 이 기술이 자율주행 시스템의 성능을 향상시키고, 안전성을 향상시키며, 사용자에게 안정적인 자율주행 경험을 제공하는 데 중요한 역할을 할 것으로 기대한다.

References

[1] Y. Wang, Z. Su, Q. Xu, and D. Fang, "Trusted and collaborative data sharing with quality awareness in autonomous driving," in *Proc. IEEE ICC*, pp. 1-6, Jun. 2021. (http://doi.org/10.1109/ ICC42927.2021.9500680)

[2] J. Byun and S. Kwon, "Effective emergency-warning message transmission in the vehicle-to-vehicle communication environment," *J. KICS*, vol. 37, no. 1, pp. 1-8, Jan. 2012. (https://doi.org/10.7840/KICS.2012.37B.1.1)

[3] "*Study on enhancement of 3GPP support for 5G V2X services* (v16.2.0, Release 16)," Dec. 2018. [Online] Available: (https://www.3gpp.org/release-16).

[4] "*Study on LTE-based V2X services* (v14.0.0, Release 14)," Jul. 2016. [Online] Available: https://www.3gpp.org/release-14.

[5] L. Liu, S. Lu, R. Zhong, et al., "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things J.*, vol. 8, no. 8, pp. 6469-6486, Apr. 2021. (http://doi.org/10.1109/JIOT.2020.3043716).

[6] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas in Commun.*, vol. 33, no. 12, pp. 2510-2523, Sep. 2015. (http://doi.org/10.1109/JSAC.2015.2478718)

[7] A. S. Kumar, L. Zhao, and X. Fernando, "Task offloading and resource allocation in vehicular networks: A lyapunov-based deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 72, no. 10, pp. 13 360-13 373, May 2023. (http://dx.doi.org/10.1109/TVT.2023.3271613)

[8] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36-44, Jun. 2017. (http://dx.doi.org/10.1109/MVT.2017.2668838)

[9] L. Kleinrock, *Queuing Systems*, 1975. (https://dl.acm.org/doi/book/10.5555/1096491)

[10] M. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Commun. Netw.*, pp. 1-211, 2010. (http://doi.org/10.2200/s00271ed1v01y201006cnt007)

최 평 준 (Pyeongjun Choi)

2020년 2월 : 대구경북과학기술원 기초학부 졸업

2020년 3월~현재 : 대구경북과학기술원 전기전자컴퓨터공학과 통합과정

<관심분야> 모바일 인공지능, 컴퓨팅/네트워킹 자원 관리

[ORCID:0000-0002-8581-8279]

윤 필 도 (Pildo Yoon)

2022년 2월 : 국립한밭대학교 컴퓨터공학과 졸업

2022년 3월~현재 : 대구경북과학기술원 전기전자컴퓨터공학과 석사과정

<관심분야> 간섭 관리, 강화 학습

[ORCID:0000-0003-2960-6308]

곽 정 호 (Jeongho Kwak)

2008년 8월 : 아주대학교 전자공학부 학사
2011년 2월 : 한국과학기술원 전기 및 전자공학과 석사
2015년 2월 : 한국과학기술원 전기 및 전자공학과 박사
2015년~2017년 : 캐나다 INRS-EMT 박사후 연구원
2017년~2019년 : 아일랜드 Trinity College Dublin Computer Science Marie-Curie Fellow
2019년 : 대구대학교 전자전기공학부 조교수
2020년~현재 : 대구경북과학기술원 조교수, 부교수
<관심분야> 6G 컴퓨팅, 스토리지, 네트워킹 자원관리, 모바일 인공지능, 저궤도 위성 엣지 컴퓨팅
[ORCID:0000-0002-5737-0665]