KIPS Trans. Comp. and Comm. Sys.
Vol.10, No.12 pp.311~318
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTSDE.2021.10.12.311

Task Migration in Cooperative Vehicular Edge Computing

Sungwon Moon† · Yujin Lim††

ABSTRACT

With the rapid development of the Internet of Things(IoT) technology recently, multi-access edge computing(MEC) is emerged as a next-generation technology for real-time and high-performance services. High mobility of users between MECs with limited service areas is considered one of the issues in the MEC environment. In this paper, we consider a vehicle edge computing(VEC) environment which has a high mobility, and propose a task migration algorithm to decide whether or not to migrate and where to migrate using DQN, as a reinforcement learning method. The objective of the proposed algorithm is to improve the system throughput while satisfying QoS(Quality of Service) requirements by minimizing the difference between queueing delays in vehicle edge computing servers(VECSs). The results show that compared to other algorithms, the proposed algorithm achieves approximately 14-49% better QoS satisfaction and approximately 14-38% lower service blocking rate.

Keywords : Task Migration, Vehicular Edge Computing, Reinforcement Learning, DQN

협력적인 차량 엣지 컴퓨팅에서의 태스크 마이그레이션

문 성 원† · 임 유 진††

요 약

최근 사물인터넷의 기술이 빠르게 발전하면서 실시간 및 고성능의 처리를 요구하는 서비스들을 위해 멀티 액세스 엣지 컴퓨팅(MEC)이 차세대 기술로 부상하고 있다. 제한적인 서비스 영역을 가지는 MEC 사이에서 사용자들의 잦은 이동성은 MEC 환경에서 다뤄야 할 문제 중 하나이다. 본 논문에서는 이동성이 많은 차량 엣지 컴퓨팅 환경(VEC)을 고려하였으며, 강화 학습 기법의 일종인 DQN을 이용하여 마이그레이션 여부와 대상을 결정하는 태스크 마이그레이션 기법을 제안하였다. 제안한 기법의 목표는 차량 엣지 컴퓨팅 서버(VECS)들의 큐잉 지연시간의 차이를 이용한 로드 밸런싱을 고려하여 QoS 만족도 향상과 시스템의 처리량을 향상시키는 것이다. 제안한 기법을 다른 기법들과의 성능 비교를 통해 QoS 만족도 측면에서 약 14-49%, 서비스 거절률 측면에서는 약 14-38%로 더 좋은 성능을 보임을 확인하였다.

키워드 : 태스크 마이그레이션, 차량 엣지 컴퓨팅, 강화 학습, DQN

1. 서 론

최근 사물인터넷(IoT)의 기술이 빠르게 발전하면서 자율주행, 증강/가상현실, 이미지 인식 등 고성능의 컴퓨팅을 요구하는 애플리케이션이 급증하고 있다[1]. 고성능의 애플리케이션은 대개 시간에 민감하고 계산 집약적이며, 이로 인하여 많은 양의 컴퓨팅 자원과 높은 QoS(Quality of Service) 보장이 요구된다. 이런 고성능의 서비스를 단말에서 제공하기는 어렵기 때문에 대게 충분한 컴퓨팅 자원이 있는 클라우드(Cloud) 서버로 오프로드(Offload) 한다.

그러나 기존 클라우드 컴퓨팅은 데이터 연산과 처리 모두 중앙에 집중되어 있기 때문에 기하급수적으로 증가하는 데이터들을 전송하고 처리하는 데 긴 응답 시간과 정체 현상이 발생할 수 있다. 이로 인해 상당한 지연시간이 발생하므로 QoS 요구 사항을 만족시키는 것이 점점 어려워진다.

그래서 이 문제를 해결하기 위해 멀티 액세스 엣지 컴퓨팅(Multi-access Edge Computing, MEC)이 차세대 기술로 부상했다[2]. 중앙 집중식 클라우드 서버에 비해, MEC 서버는 사람 및 사물과 근접한 네트워크 종단에 컴퓨팅 자원을 분산 배치한다. 사용자(단말)와 지리적으로 근접하기 때문에 오프로드된 작업을 바로 처리하고, 빠른 연산이 가능하기 때문

※ 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2021R1F1A1047113).
※ 이 논문은 2021년 한국정보처리학회 춘계학술발표대회에서 “차량 엣지 컴퓨팅에서 로드 밸런싱을 고려한 강화학습 기반의 마이그레이션”의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 숙명여자대학교 IT공학과 석·박사통합과정
†† 종신회원 : 숙명여자대학교 IT공학과 교수
Manuscript Received : July 1, 2021
First Revision : August 25, 2021
Accepted : September 9, 2021
* Corresponding Author : Yujin Lim(yujin91@sookmyung.ac.kr)

※ This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/ licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

에 QoS 요구 사항 만족도가 증가할 뿐만 아니라 작업의 신뢰성을 향상시킬 수 있다.

MEC 시스템에서 다뤄야 할 문제 중 하나는 이동성이다. 사용자가 현재 서비스 중인 MEC 서버의 영역 밖으로 이동하였을 때 시스템은 현재 MEC 서버에서 이동 대상인 MEC 서버로 태스크 마이그레이션(migration)을 수행해야 한다. 특히, 이동성이 많은 환경에서 효율적인 태스크 마이그레이션은 QoS 요구 사항 만족에 아주 중요한 고려사항이다. 제한적인 서비스 영역을 가지는 MEC 서버들 사이에서 사용자의 높은 이동성은 서비스 중단과 QoS 저하를 초래하기 때문이다. 그래서 이동성에 맞게 MEC 서버 간 마이그레이션 진행 여부와 대상을 결정하는 태스크 마이그레이션이 중요하다[3].

사용자들의 높은 이동성으로 인해 특정 MEC 서버만 혼잡해지는 경우가 발생할 수 있으며, 이는 MEC 서버들의 로드 불균형을 초래한다. MEC 서버들의 로드 불균형은 MEC 시스템의 처리량이 낮아질 뿐만 아니라 MEC 서버들의 큐잉 지연시간이 길어져 낮은 QoS를 초래할 수 있다.

그래서 이러한 문제는 MEC 서버들 간 로드 밸런싱을 통해 특정 MEC 서버만 혼잡해지는 경우를 줄임으로써 시스템의 처리량을 향상시켜 서비스의 QoS 만족도도 향상시킬 수 있다[4]. 그러나 마이그레이션을 진행하는 데 필요한 시간과 컴퓨팅 자원에 대한 추가 비용이 필요하다. 따라서 본 논문에서는 이동성이 높은 차량 엣지 컴퓨팅(Vehicular Edge Computing, VEC) 환경에서 차량 엣지 컴퓨팅 서버 (VECS, VEC Server)들의 큐잉 지연시간을 이용한 로드 밸런싱을 하면서 마이그레이션 비용을 최소화하기 위한 효율적인 태스크 마이그레이션 전략을 제안한다. 그리고 본 논문은 협력적인 VECS 그룹 환경을 채택하여 그룹 내 VECS들과 협업할 수 있는 환경을 가정하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 소개하고, 3장에서는 시스템 모델에 대해 설명한다. 4장에서는 본 논문에서 제안하는 알고리즘을, 5장에서는 실험 및 결과 분석을, 마지막으로 6장에서는 결론 및 향후 연구를 다룬다.

2. 관련 연구

태스크 마이그레이션은 사용자의 높은 이동성과 고성능을 요구하는 애플리케이션으로 인해 발생하는 QoS 저하와 길어지는 지연시간 문제를 효율적으로 해결할 수 있는 해결책으로 인식되고 있다. 높은 이동성과 제한적인 서비스 영역을 가지는 MEC 서버로 인해 태스크 마이그레이션에 관한 연구가 많이 진행되고 있다.

[5]에서는 시스템의 최대 이득을 얻기 위해 Q-learning을 기반으로 한 태스크 오프로딩 및 마이그레이션 기법을 다루었다. 태스크의 속성, 사용자의 이동성 및 사용자의 셀 체류시간을 기반으로 오프로딩 및 마이그레이션 결정하는 기법을 제안하였다. 이 외에도 기존의 핸드오버 기법을 사용하며, 실시간 서비스를 위해 QoS 기반의 마이그레이션 여부를 결정하는 기법[6] 또한 제안되었다. [5,6]은 마이그레이션 대상으로는 사용자의 이동 대상인 MEC 서버이며, 해당 MEC 서버로 마이그레이션 진행 여부만을 결정하는 기법들이다.

마이그레이션 진행 여부뿐만 아니라 대상도 결정하는 마이그레이션 기법도 연구되고 있다. [7]에서는 차량의 이동성과 지연 제약조건을 고려하면서 전체 시스템 비용을 최소화하기 위해 최적의 MEC를 선택하는 기법이다. 이때, 차량의 위치에서 근접한 모든 MEC 서버가 대상이 된다. 본 논문에서는 협력적인 VECS 그룹 내 모든 VECS가 마이그레이션 대상이 되지만, [7]은 근접한 서버만 대상이 되는 점이 차이다. [8]에서는 마이그레이션을 진행시 사용하는 통신 비용과 마이그레이션 비용을 최소화하기 위한 마이그레이션 여부 및 대상을 결정하는 기법을 제안하였다. 이는 마이그레이션 비용을 기반으로 마이그레이션 여부 및 대상을 결정하였지만, 본 논문은 마이그레이션 비용뿐만 아니라 서버의 상태도 고려한 마이그레이션 기법을 제안하였다.

높은 이동성 또는 많은 사용자로 인해 MEC 서버들 사이의 로드 불균형이 발생할 수 있다. 이는 QoS 저하와 연산 지연시간을 초래한다. 그래서 현재 로드 밸런싱을 고려한 오프로딩 및 마이그레이션 기법들이 연구되고 있다.

[9]에서는 단말과 MEC 서버의 로드 밸런싱을 고려한 오프로딩 기법을 제안하였다. 태스크의 지연시간을 최소화하기 위하여 MEC 서버의 컴퓨팅 자원을 활용하여 로드 밸런싱을 고려하였다. 그리고 [10]에서는 유틸리티 함수를 이용하여 로드 밸런싱을 고려한 오프로딩 기법이 제안되었다. 단말과 MEC 서버에서의 서비스 지연시간을 이용하여 QoS 기반의 유틸리티 함수를 설계하여 에너지 소비를 최소화하는 기법을 제안하였다. [9,10]은 단말과 MEC 서버 사이의 로드 밸런싱만 고려하고, MEC 서버들 사이는 고려하지 않았다는 한계점이 있다. 또한, [11]은 서버의 로드와 서버의 네트워크 로드의 편차를 이용하여 로드 밸런싱을 고려하여 최적의 서버로 오프로딩하는 기법을 제안하였다. [12]는 VECS 서버들 간의 워크로드의 편차를 이용하여 로드 밸런싱을 고려한 태스크 마이그레이션 기법을 제안하였다. [11]은 자원의 활용률로, [12]는 태스크의 연산 작업량으로 서버의 로드를 정의하였으며, 본 논문은 서버의 큐잉 지연시간으로 정의하였다. VECS들의 로드 차이를 이용하여 로드 밸런싱을 고려했다는 점에서는 같지만, 본 논문은 로드 밸런싱 뿐만 아니라 태스크가 지연제약 조건에 만족함에도 불구하고 발생할 수 있는 불필요한 마이그레이션을 방지하고자 마이그레이션 비용에 대해서도 고려했다는 점에서 차이가 있다.

본 연구에서는 차량 엣지 컴퓨팅 환경에서 차량의 이동성에 맞게 강화 학습 기법인 Deep Q-learning(DQN)[13]을 적용

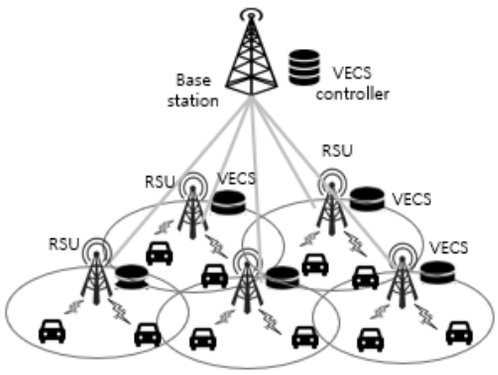

Fig. 1. Vehicular Edge Computing Model

한 태스크 마이그레이션 기법을 제안한다. 본 기법은 VECS들의 로드 밸런싱과 마이그레이션 비용을 고려하여 서비스의 QoS 만족도와 시스템의 처리량을 향상시키고자 제안하였다.

3. 시스템 모델

Fig. 1처럼 본 논문에서 제안된 시스템은 M개의 VECS가 각 RSU(Road Side Unit)에 배치되었으며, 이는 하나의 협력적인 VECS 그룹을 구성하고 있다고 가정한다. VECS들은 서로 백홀(Backhaul) 채널을 통해 연결되며, 각 VECS들은 $M/M/1$ 큐를 모델화했다. 그리고 각 VECS들은 VECS 그룹 컨트롤러와 연결되어 있다. VECS 컨트롤러는 차량과 그룹 내 VECS들에 대한 정보를 모두 수집하며, 마이그레이션 여부 및 대상에 관한 결정을 한다. 마이그레이션 결정 시점은 기존의 핸드오버 기법을 사용하여, 차량이 인접한 VECS들 간 서비스 커버리지가 중첩되어 있는 곳에 위치하게 될 때 이뤄진다. N개의 차량들이 도로를 주행하며, 차량에서 발생된 태스크는 해당하는 VECS로 오프로드 한다. 차량에서 발생한 태스크 $n \in N$은 $s_n = \{i_n, \lambda_n, T_n^{\max}\}$로 정의되며, i_n는 태스크의 사이즈, λ_n는 태스크를 처리하기 위해 요구되는 CPU 사이클이고, $T_n^{\max}$는 태스크의 최대 허용 가능한 지연시간이다. 각 차량은 하나의 태스크를 발생시키며, 태스크는 하나의 VECS에서만 수행할 수 있다. 차량 n과 VECS $m \in M$ 사이의 전송 속도는 다음과 같다.

$$R_{n,m} = B_n \log_2 (1 + \frac{P_n H_{n,m}}{\sigma^2}) \tag{1}$$

B_n는 전송 대역폭, P_n는 차량의 송신전력 그리고 σ^2는 잡음이다. $H_{n,m}$는 차량 n과 VECS m사이의 채널 이득이다. 차량 n의 태스크가 VECS m으로 오프로드 될 때, 전송 지연시간은 다음과 같다.

$$T_n^{comm} = \frac{i_n}{R_{n,m}} \tag{2}$$

태스크의 연산 결과는 태스크의 입력 크기보다 훨씬 작아 다운로드 지연시간은 고려하지 않고 업로드 지연시간만 고려한다. 만약 차량 n이 VECS m에서 태스크를 수행한다면, VECS m의 큐잉 지연시간은 다음과 같다.

$$T_m = \frac{1}{C_m - \sum_{n=1}^{N} x_{n,m} \lambda_n} \tag{3}$$

여기서, C_m은 VECS m의 컴퓨팅 능력이고, $x_{n,m}$는 차량 n의 태스크가 VECS m에서 수행 중인지의 여부를 나타내는 이진 변수이다. VECS m으로 오프로드된 차량 n의 태스크를 수행하는데 소요되는 지연시간은 다음과 같다.

$$T_n^{comp} = T_m + \frac{\lambda_n}{C_m} \tag{4}$$

여기서, T_m은 태스크 n이 VECS m에 배정된 시간부터 큐에서 대기하는 시간이고, λ_n / C_m은 태스크 n이 VECS m에서 연산되는 시간이다. 마이그레이션 지연시간은 백홀 채널을 통해 VECS m에서 VECS m'로 전송되는 지연시간이며, 다음과 같다.

$$T_n^{mig} = \begin{cases} 0, & m = m' \\ \dfrac{i_n}{R_b}, & m \neq m' \end{cases} \tag{5}$$

여기서 R_b는 백홀 채널의 전송 속도이다. 따라서 태스크의 총 지연시간은 다음과 같다.

$$T_n^{total} = T_n^{comm} + T_n^{comp} + T_m^{mig} \tag{6}$$

본 논문에서의 목표는 VECS들 사이의 큐잉 지연시간의 차이를 최소화하면서 QoS 만족도와 시스템의 처리량을 향상시키는 것이다. VECS들 사이 태스크가 얼마나 적절하게 분산되었는지 측정하기 위해 각 VECS들의 큐잉 지연시간의 차이를 이용한다.

$$QD^{diff} = \sum_{m=1}^{M} \sum_{m' \neq m}^{M} |T_m - T_{m'}| \tag{7}$$

마이그레이션이 진행될 때 VECS m에서 VECS m'으로 연산 복제 비용과 VECS m의 자원 릴리즈와 같은 추가적인 비용을 발생된다[14]. 그래서 차량 n의 마이그레이션 비용은 태스크의 사이즈에 따라 결정되며, 다음과 같다.

$$MG_n = \begin{cases} 0, & m = m' \\ \mu |i_n|, & m \neq m' \end{cases} \tag{8}$$

여기서 μ는 양의 계수이다. MG는 모든 차량들의 태스크들의 MG_n의 총합이며, 다음과 같다.

$$MG = \sum_{n=1}^{N} MG_n \tag{9}$$

본 논문의 목표는 큐잉 지연시간을 고려한 로드 밸런싱을 최적화시킴으로써 전체적인 QoS를 향상시키는 것이다. 그리고 마이그레이션 비용을 최소화하는 것이다. 두 목표 각각 0-1 사이의 값으로 정규화하고, 가중치 $w \in [0,1]$를 사용하여 가중치 합계로 정의된다. 결과적으로, 식은 다음과 같이 표현 가능하다.

$$min \quad wQD^{diff} + (1-w)MG \tag{10}$$

$$s.t. \quad T_n \tag{10a}$$

제약조건 (10a)는 차량에서 발생한 태스크의 총 지연시간 (6)이 태스크의 허용 가능한 최대 지연시간($T_n^{\max}$)을 만족해야 한다. 차량과 VECS의 개수가 증가할수록 문제의 크기가 점점 커진다는 문제점이 있다. 기존의 최적화 기법 대신, 본 논문은 강화 학습 기법인 DQN을 사용하여 이 문제를 해결하고자 한다.

4. 제안하는 알고리즘

본 논문에서는 강화 학습인 DQN을 기반으로 마이그레이션 여부 및 대상을 결정하는 알고리즘을 제안한다. VECS 컨트롤러는 에이전트 역할을 한다. VECS들뿐만 아니라 차량들의 정보들을 관찰하여 서비스의 QoS를 만족하면서 전체 시스템의 처리량을 높이고 마이그레이션 비용을 최소화하는 마이그레이션 결정을 내릴 수 있다.

Q-learning은 Q 함수(Q-function)을 사용하여 최적의 행동을 선택하는 강화 학습의 일종으로 상태(State), 행동(Action) 그리고 보상(Reward)로 구성되어 있다. 최적의 정책을 찾기 위해 Q 테이블(Q-table)에 저장된 Q 값(Q value)이 가장 큰 행동을 선택하는 Q 함수는 다음과 같다.

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_a Q(s',a') - Q(s,a)] \tag{11}$$

현재 상태(s), 행동(a), 보상(r) 그리고 새로운 상태(s')와 행동(a')에 대한 최댓값을 구한다. α는 학습률(Learning rate)이고, γ는 할인율(Discount factor)이다. 최적의 Q 함수는 다음과 같이 표현 가능하다.

$$Q^*(s,a) = \max_a Q(s',a') \tag{12}$$

그러나 상태, 행동 공간의 차원이 높은 경우 Q-learning의 효율성이 저하된다. 이 문제를 해결하기 위해, 신경망을 활용하여 각 활동의 보상을 예측하는 DQN 기반의 마이그레이션 기법을 제안한다. DQN은 상태 공간이 크고 복잡한 고차원 환경에 더욱 적합하다.

Table 1. Deep Q-Learning Algorithm

Algorithm 1. Deep Q-learning Algorithm

Initialize main DQN $Q(s,a;\theta)$ with random weightsθ
Initialize target DQN $Q(s,a;\theta')$ with weights $\theta' = \theta$
Initialize replay memory D to capacity N
for each episode **do**
  Initialize initial state s_0, reward r_0
  for time slot $t = \tau, 2\tau, ..., T$ **do**
    Controller acquires information about vehicles, tasks and VECS by interacting with the environment
    If the random number $< \epsilon$:
      Select action $a_t = \arg max_a Q(s_t, a_t)$
    else:
      Randomly select action a_t
    Execute action a_t at controller, observe reward r_t and next state s_{t+1}
    Store the tuple $< s_t, a_t, r_t, s_{t+1} >$ in D
    Randomly sample a mini-batch of tuple $\{< s_t, a_t, r_t, s_{t+1} >\}_{j=1}^{J}$ from D
    If episode terminates at $j+1$ then
      $y_j = r_j$
    else:
      $y_j = r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta')$
    Perform a gradient descent step on $Loss(\theta)$ with respect to the network parametersθ, where loss function is $Loss(\theta) = 1/J \sum [y_j - Q(s_j, a_j; \theta)]^2$ and update $\theta' = \theta$
    Terminate when all the vehicles are out of VECS
  end for
end for

DQN은 신경망을 사용하여 $Q^*(s,a)$를 학습시킨다. DQN의 Q함수는 $Q^*(s,a;\theta)$로 정의되며, θ는 메인 DQN의 가중치를 의미한다. 네트워크는 손실함수 $Loss(\theta)$을 최소화하면서 학습되며, 다음과 같다.

$$Loss(\theta) = E[y - Q(s,a;\theta)]^2 \tag{13}$$

여기서 $y = r + \gamma \max_{a'} Q(s',a';\theta')$는 타겟 Q값이고, θ'는 θ를 주기적으로 갱신한다. DQN은 다음 Table 1과 같다.

본 논문에서 정의한 상태(s), 행동(a) 그리고 보상(r)의 정의는 다음과 같다. VECS들의 큐잉 지연시간을 이용하여 로드 밸런싱을 최적으로 하기 위해 차량이 발생한 태스크 정보와 해당 태스크를 현재 수행 중인 VECS(l_n) 그리고 마이그레이션 결정 여부에 대한 이진 정보(o_n)를 반영해야 한다. 따라서 상태는 $S = \{i_n, \lambda_n, l_n, o_n\}$으로 표현 가능하다.

VECS 컨트롤러는 현재 상태(s)를 기반으로 행동(a)을 선택하

고, 해당 행동을 취한다. 행동은 마이그레이션 진행 여부 및 대상을 결정할 수 있도록 정의하였다. 행동$A=\{a_1,a_2,...,a_M\}$으로 표현 가능하다. 만약 a_n이 지금 서비스 받고 있는 VECS(l_n)와 같다면 마이그레이션을 진행하지 않고, 다르다면 a_n으로 마이그레이션을 진행한다.

VECS 컨트롤러는 행동(a)을 취함으로써 보상(r)을 얻는다. 본 논문에서 제안한 알고리즘의 목표인 QoS 만족도와 시스템 처리량을 향상시키기 위해 식 (10)처럼 VECS들의 큐잉 지연시간 차이와 마이그레이션 비용을 최소화하는 방향으로 학습을 진행한다. 따라서 부정적인 보상으로 $r=-(wQD^{diff}+(1-w)MG)$와 같이 표현 가능하다.

5. 실험 및 결과 분석

제안한 마이그레이션 알고리즘에 대한 성능을 측정하기 위해 실험을 진행하였다. 총 5개의 RSU에 각 VECS가 배치된 환경을 고려하였다. RSU의 반경은 250m이고, 각 VECS의 컴퓨팅 능력은 10GHZ이다. 차량은 푸아송 분포를 이용하여 발생시켰다. 태스크의 사이즈 i_n는 [100,150]MB, 이에 요구되는 CPU 사이클 λ_n는 [40,75]Gycles 범위 내에서 무작위로 발생시켰고, $T_n^{\max}$는 60s다. B_n은 10MHZ, P_n은 1W이고, σ^2은 10^{-11}mW 그리고 R_b는 20MHZ이다. μ는 0.002, ω는 0.5 그리고 타임 슬롯의 길이는 5s로 설정하였다[15]. 본 실험은 샌프란시스코의 이동성 데이터를 사용하였다[16]. DQN을 위해 α는 0.01, γ는 0.9, ϵ는 0.9로 설정하였다. 리플레이 메모리의 용량은 500, 미니배치의 사이즈는 32로 설정하였다.

본 논문에서 제안한 DQN 기반의 마이그레이션 기법인 *Proposed*의 성능 비교를 위해 No *Migration, Nearest, Random, Compare1* 그리고 *Compare2* 총 5가지 기법을 비교하였다. *No Migration*은 차량의 이동과 상관없이 처음에 오프로드된 VECS에서 태스크가 완료될 때까지 마이그레이션을 진행하지 않는 기법이다. *Nearest*는 차량의 이동에 따라 지리적으로 가장 근접한 VECS로 마이그레이션을 항상 진행하는 기법이다. *Random*은 마이그레이션 진행 여부와 대상을 랜덤으로 결정하는 기법이다. *Compare1*은 VECS들의 워크 로드의 편차를 이용한 로드밸런싱을 고려한 마이그레이션 결정 기법[10]이다. *Compare2*는 마이그레이션 진행 시 사용되는 네트워크에 대한 비용과 마이그레이션 비용을 고려하여 마이그레이션을 결정하는 기법이다[6].

Fig. 2는 강화 학습 모델 훈련 성능 평가에 관한 실험이다. 강화 학습이 적절히 작동하였는지 보기 위해 에피소드에 따른 에피소드 리워드를 분석하였다. 본 논문에서 정의한 리워드는 부정적인 보상이므로 리워드 값이 작을수록 좋다. DQN 모델을 200번 동안 학습하였으며, 에피소드가 반복될수록 본 논문의 목표인 리워드가 감소하였으며, 리워드의 값은 에피소드 150 이후 안정화되는 것을 볼 수 있다. 이 의미는 제안한 기법의 강화 학습 정책이 최적으로 작동한다는 것을 보여준다.

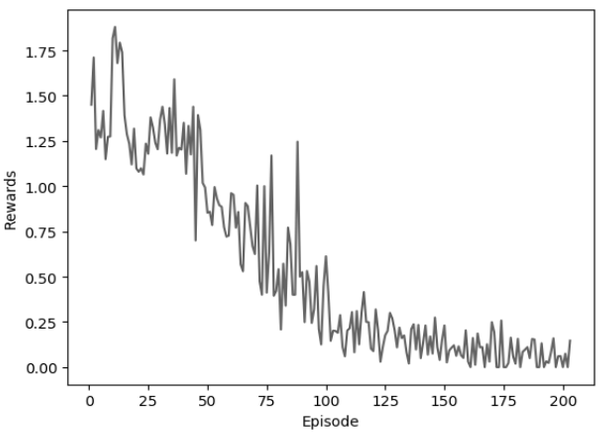

Fig. 2. Comparison of the Rewards for Episode

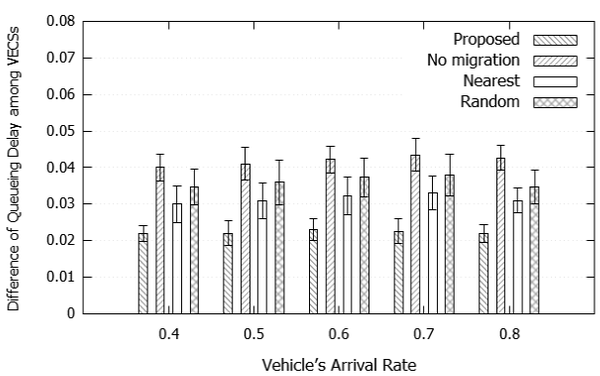

Fig. 3. Comparison of the Queueing Delay among VECSs for Different Arrival Rates

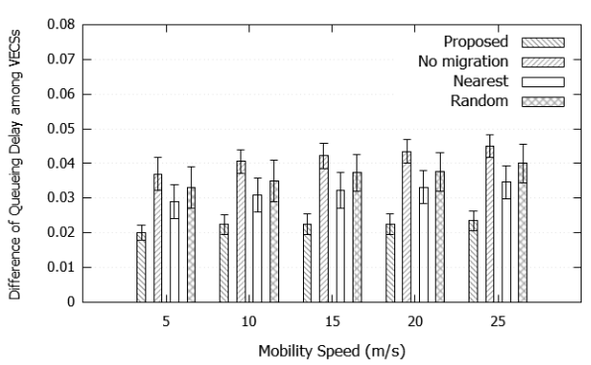

Fig. 4. Comparison of the Queueing Delay among VECSs for Different Mobility Speeds

Fig. 3과 4는 차량의 도착률과 속도에 따른 VECS들의 큐잉 지연시간들에 대한 차이를 보여주는 실험이다. Fig. 3과 4를 보듯이 마이그레이션을 진행하는 것이 진행하지 않는 경우보다 VECS들의 큐잉 지연시간의 차이가 작다는 것을 볼 수 있다. 이는 특정 VECS로 치우치는 경향이 덜 한다는 의미로 로드 밸런싱이 적절하게 되었다는 것이다.

Fig. 3에서 *Proposed* 기법을 *No Migration, Nearest,* 그리고 *Random* 기법과 비교하였을 때 약 47%, 30% 그리고 38%만큼 차이가 난다. *No Migration*이 성능이 가장 안 좋은 이유는 푸아송 분포에 따라 차량을 각 VECS 반경에 분포했기 때문에 특정 VECS로 몰리는 경향이 더 크기 때문이다. *Nearest*는 마이그레이션을 항상 진행하므로 *No Migration* 보다는 성능이 좋지만, *Proposed*보다는 안 좋다. 가장 가까운 곳으로 마이그레이션을 진행하기 때문에 특정 VECS로 몰리는 경우도 발생하기 때문이다. *Random*은 마이그레이션 여부를 랜덤으로 정하기 때문에 마이그레이션을 진행하는 경우도 있어 *No Migration*보다는 성능이 좋지만, 진행하지 않는 경우도 있고 대상을 랜덤으로 정하기 때문에 *Nearest*보다는 성능이 안 좋다. 그리고 차량의 도착률이 증가할수록 큐잉 지연시간의 차이가 커지다가 도착률이 높아진 경우에는 VECS들 대부분 꽉차있기 때문에 차이가 다시 줄어드는 것을 확인할 수 있다.

Fig. 4에서는 *Proposed* 기법을 *No Migration, Nearest,* 그리고 *Random* 기법과 비교하였을 때 약 46%, 31%, 39%만큼 차이가 난다. 차량의 속도가 높다는 것은 그만큼 차량의 이동성이 높다는 것이 의미하며, 이는 마이그레이션을 결정할 기회가 많다는 것을 의미한다. 그렇기 때문에 VECS 큐잉 지연시간을 고려하여 로드 밸런싱을 할 기회가 많아진다. 그래서 속도가 높아질수록 큐잉 지연시간의 차이가 증가하는 것을 볼 수 있다.

Fig. 5와 6에서는 차량의 도착률과 속도에 따른 서비스 거절률(Blocking)을 보여주는 실험이다. 여기서, 서비스 거절률은 큐가 꽉 차서 거절되는 수를 의미하며, 이를 거절된 수를 전체 태스크 수로 나누어 정의하였다. Fig. 5에서 *Proposed* 기법을 *No Migration, Nearest,* 그리고 *Random* 기법과 비교하였을 때 약 49%, 38%, 43%만큼 차이가 났으며, *Compare1*과 *Compare2* 기법과 비교하였을 때, 약 17%, 31%만큼 더 좋은 성능을 보인다. Fig. 6에서는 *Proposed* 기법을 *No Migration, Nearest, Random, Compare1* 그리고 *Compare2* 기법과 비교하였을 때, 약 48%, 34%, 41%, 14% 그리고 30% 만큼 성능 차이가 난다.

No Migration은 특정 VECS로 몰리는 경우가 많아 거절률이 높고, 그다음으로는 *Random*이 높다. 그 이유로는 마이그레이션을 진행하지 않는 경우도 있으며, 대상도 랜덤으로 선정하기 때문에 이미 거절률이 높은 VECS를 선택하는 경우도 많으며, 특정 VECS만 선택하는 경우도 발생하기 때문이다. 이에 비하면 근접한 VECS로 진행하는 *Nearest*도 한쪽으로 치우치는 경향이 있지만 이는 차량의 경로에 따라 달라지는 경우도 있기 때문에 *No Migration, Random*보다는 성능이 좋다. VECS의 워크로드 편차를 이용하여 VECS의 상태를 고려하여 마이그레이션 여부와 대상을 결정했기 때문에 *Compare1*이 *Proposed* 다음으로 성능이 가장 좋다. 그

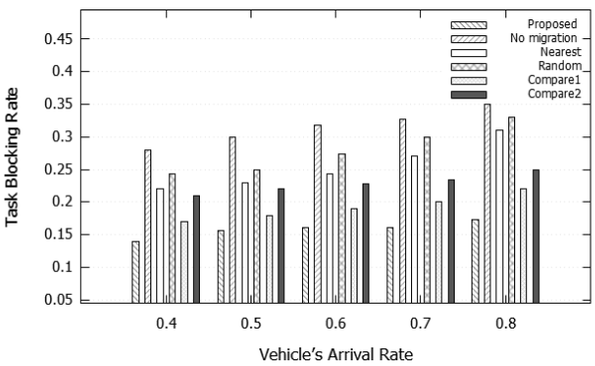

Fig. 5. Comparison of the Task Blocking Rate for Different Arrival Rates

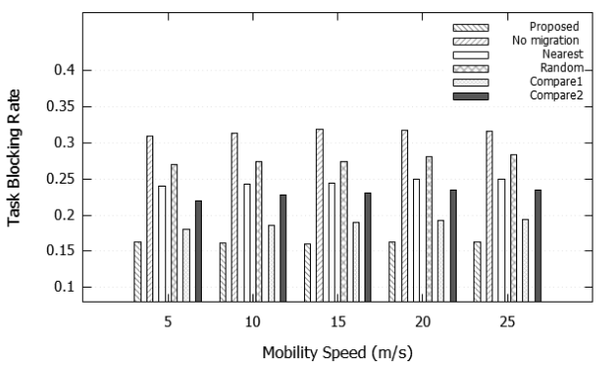

Fig. 6. Comparison of the Task Blocking Rate for Different mobility Speeds

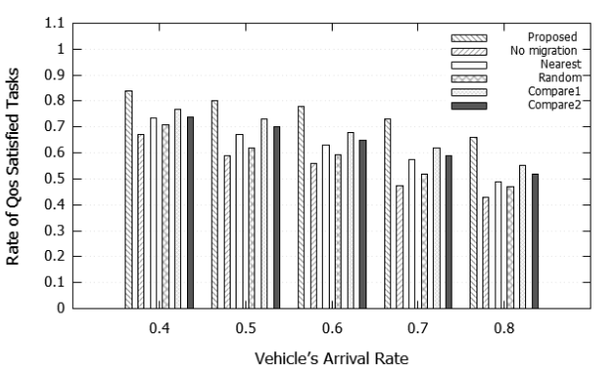

Fig. 7. Comparison of the QoS Satisfied Rate for Different Arrival Rates

리고 VECS의 상태보다는 마이그레이션 진행시 드는 네트워크 비용과 마이그레이션 비용을 고려한 *Compare2*가 그다음으로 성능이 좋다. *Compare2*는 VECS들의 로드 밸런싱을 고려하기보다는 마이그레이션과 네트워크 비용에 더 치중했기 때문에 *Proposed*와 *Compare1*과 성능 차이가 난다.

Fig. 7과 8에서는 차량의 도착률과 속도에 따른 QoS 만족도를 보여주는 실험이다. 여기서 QoS 만족도는 지연시간 내 완료한 태스크 수를 전체 태스크 수로 나누어 정의하였다. Fig. 7에서 *Proposed* 기법이 *No Migration, Nearest,*

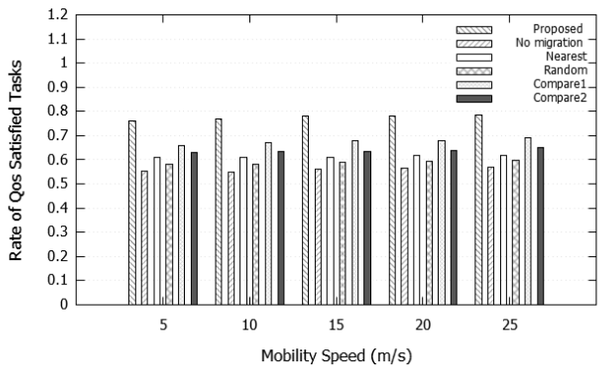

Fig. 8. Comparison of the QoS Satisfied Rate for Different Mobility Speeds

Random, Compare1 그리고 *Compare2* 기법과 비교하였을 때, 약 32%, 23%, 25%, 14% 그리고 20%만큼 더 좋은 만족도를 보인다. Fig. 7에서는 각각 약 38%, 26%, 30%, 15% 그리고 21%만큼 더 좋은 만족도를 보인다. 로드밸런싱을 고려한 *Compare1*이 *Proposed* 다음으로 QoS 만족도가 높았고, 그다음으로는 마이그레이션 비용을 고려한 *Compare2*가 높았다. Fig. 3,4와 Fig. 5,6에서 보였듯이 큐잉 지연시간의 차이를 통해 로드 밸런싱을 하면 VECS들의 서비스 거절률이 낮아져 시스템의 처리량이 높아지며, 이로 인해 QoS 만족도가 높아지는 것을 확인할 수 있다.

6. 결 론

차량 엣지 컴퓨팅 환경에서 차량들의 높은 이동성으로 인해 특정 MEC 서버만 혼잡해지는 경우가 있어 MEC 서버들의 로드 불균형이 발생된다. 이는 MEC 시스템의 처리량이 낮아질 뿐만 아니라 MEC 서버들의 큐잉 지연시간이 길어져 낮은 QoS를 초래한다. 이를 해결하고자 본 논문은 강화 학습 기법인 DQN을 이용하여 태스크의 VECS 마이그레이션 여부 및 대상을 결정하는 기법을 제안하였다. 제안한 기법은 QoS 만족도와 시스템의 처리량을 향상시키기 위해서 VECS들의 큐잉 지연시간을 이용한 로드 밸런싱을 최적화하고 마이그레이션 비용을 최소화하는 것이다. 실험을 통해 VECS들 사이의 큐잉 지연시간 차이와 서비스 거절률 그리고 QoS 만족도에서 다른 기법들에 비해 우수한 성능을 보임을 확인하였다. 하지만 본 논문은 멀티 유저 환경에서 싱글 에이전트를 고려한 상황이며, 협력적인 VECS 그룹의 개수를 고정시킨 환경이다. 보다 동적인 환경에 있어 효율적으로 학습하기 위해서는 멀티 에이전트를 고려한 마이그레이션 연구가 필요하다.

References

[1] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5G-enabled tactile internet," *IEEE Journal on Selected Areas in Communications*, Vol.34, No.3, pp.460-473, 2016.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, Vol. 19, No.4, pp.2322-2358, 2017.

[3] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, Vol.6, pp.23511-23528, Apr. 2018.

[4] D. Baburao, T. Pavankumar, and C. S. R. Prabhu, "Survey on service migration, load optimization and load balancing in fog computing environment," *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pp.1-5, India, 2019.

[5] D. Wang, X. Tian, H. Cui, and Z. Liu, "Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware MEC network," China Communications, Vol.17, No.8, pp.31-44, 2020.

[6] J. Li, X. Shen, L. Chen, D. Pham, J. Ou, L. Wosinska, and J. Chen, "Service migration in fog computing enabled cellular networks to support real-time vehicular communications," *IEEE Access*, Vol.7, pp.13704-13714, 2019.

[7] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, Vol.7, pp.26652-26664, 2019.

[8] Z. Gao, Q. Jiao, K. Xiao, Q. Wang, Z. Mo, and Y. Yang, "Deep reinforcement learning based service migration strategy for edge computing," *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, San Francisco, pp.116-1165, 2019.

[9] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, Vol.69, No.2, pp.2092-2104, 2020.

[10] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, Vol.6, No.3, pp.4377-4387, 2019.

[11] K. Addali and M. Kadoch, "Enhanced mobility load balancing algorithm for 5G small cell networks," *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, Canada, pp.1-5, 2019.

[12] S. Moon and Y. Lim, "Migration with balancing based on reinforcement learning in vehicular edge computing," *KIPS Conference 2021*, Korea, May 2021.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, l. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *NIPS Deep Learning Workshop 2013*, Dec. 2013.

[14] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Transactions on Vehicular Technology*, Vol.69, No.8, pp.9041-9052, 2020.

[15] C. Liu, F. Tang, Y. Hu, K. Li, Z. Tang, and K. Li, "Distributed task migration optimization in MEC by extending multi-agent deep reinforcement learning approach," *IEEE Transactions on Parallel and Distributed Systems*, Vol.32, No.7, pp.1603-1614, Jul. 2021.

[16] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, CRAWDAD DataSet Epfl/Mobility, Feb. 2009. Available: http://crhttp://crawdad.org/epfl/mobility (accessed on 24 August 2021).

문 성 원

https://orcid.org/0000-0003-1255-0387
e-mail : sungwon268@sookmyung.ac.kr
2019년 숙명여자대학교 IT공학과(학사)
2019년 ~ 현 재 숙명여자대학교 IT공학과 석 · 박사통합과정
관심분야 : 지능형 시스템, IoT, Edge Computing

임 유 진

https://orcid.org/0000-0002-3076-8040
e-mail : yujin91@sookmyung.ac.kr
2000년 숙명여자대학교 전산학과(박사)
2013년 일본 Tohoku University, Department of Information Sciences(박사)
2004년 ~ 2015년 수원대학교 정보미디어학과 부교수
2016년 ~ 현 재 숙명여자대학교 IT공학과 교수
관심분야 : 지능형 시스템, IoT, Edge Computing