**KIPS Trans. Comp. and Comm. Sys.**
Vol.11, No.5 pp.139~146
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)


https://doi.org/10.3745/KTCCS.2022.11.5.139

# Performance Comparison of Task Partitioning Methods in MEC System

Sungwon Moon† · Yujin Lim††

## ABSTRACT

With the recent development of the Internet of Things (IoT) and the convergence of vehicles and IT technologies, high-performance applications such as autonomous driving are emerging, and multi-access edge computing (MEC) has attracted lots of attentions as next-generation technologies. In order to provide service to these computation-intensive tasks in low latency, many methods have been proposed to partition tasks so that they can be performed through cooperation of multiple MEC servers(MECSs). Conventional methods related to task partitioning have proposed methods for partitioning tasks on vehicles as mobile devices and offloading them to multiple MECSs, and methods for offloading them from vehicles to MECSs and then partitioning and migrating them to other MECSs. In this paper, the performance of task partitioning methods using offloading and migration is compared and analyzed in terms of service delay, blocking rate and energy consumption according to the method of selecting partitioning targets and the number of partitioning. As the number of partitioning increases, the performance of the service delay improves, but the performance of the blocking rate and energy consumption decreases.

Keywords : MEC, Task Partitioning, Task Offloading, Task Migration

# MEC 시스템에서 태스크 파티셔닝 기법의 성능 비교

문 성 원† · 임 유 진††

## 요 약

최근 사물 인터넷의 발전과 함께 차량과 IT 기술의 융합되어 자율주행과 같은 고성능의 어플리케이션들이 등장하면서 멀티 액세스 엣지 컴퓨팅(MEC)이 차세대 기술로 부상하였다. 이런 계산 집약적인 태스크들을 낮은 지연시간 안에 제공하기 위해, 여러 MEC 서버(MECS)들이 협력하여 해당 태스크를 수행할 수 있도록 태스크를 파티셔닝하는 기법들이 많이 제안되고 있다. 태스크 파티셔닝과 관련된 연구들은 모바일 디바이스에서 태스크를 파티셔닝하여 여러 MECS들에게 오프로딩을 하는 기법과 디바이스에서 MECS로 오프로딩한 후 해당 MECS에서 파티셔닝하여 다른 MECS들에게 마이그레이션하는 기법으로 나누어볼 수 있다. 본 논문에서는 오프로딩과 마이그레이션을 이용한 파티셔닝 기법들을 파티셔닝 대상 선정 방법 및 파티셔닝 개수 변화에 따른 서비스 지연시간, 거절률 그리고 차량의 에너지 소비량 측면에서의 성능을 분석하였다. 파티셔닝 개수가 증가할수록 지연시간의 성능은 향상하나, 거절률과 에너지 소모량의 성능은 감소한다.

키워드 : MEC, 태스크 파티셔닝, 태스크 오프로딩, 태스크 마이그레이션

## 1. 서 론

최근 사물 인터넷(Internet of Things, IoT)의 발전과 함께 차량과 IT 기술의 융합이 가속화되면서 기존의 차량 통신 네트워크가 차량 인터넷(Internet of Vehicles, IoV)으로 확장되어 자율주행, 차량 안전 서비스, 경로 계획과 같은 다양한 고성능의 차량 어플리케이션들이 등장했다[1,2]. 이러한 어플리케이션들은 계산 집약적이며, 높은 신뢰성과 낮은 지연시간 등의 QoS(Quality of Service)를 요구한다. 하지만 차량의 컴퓨팅 성능이 많이 향상되었음에도 불구하고, 이를 차량에서 제공하기에는 한계가 있다.

이를 극복하고 낮은 지연시간으로 서비스를 제공받기 위해, 차량은 발생한 태스크를 클라우드(cloud) 서버로 오프로드(offload) 한다. 최근 들어 지리적으로 원거리에 위치한 기존 중앙 집중형 클라우드 컴퓨팅과 달리, 사람 및 사물과 근접한 네트워크 종단에 컴퓨팅 자원을 분산 배치한 멀티 액세

※ 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2021R1F1A1047113).
※ 이 논문은 2021년 한국정보처리학회 ACK 2021의 우수논문으로 "MEC 환경에서 오프로딩과 마이그레이션을 이용한 태스크 파티셔닝 기법의 성능비교"의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 숙명여자대학교 IT공학과 석・박사통합과정
†† 종신회원 : 숙명여자대학교 IT공학과 교수
Manuscript Received : December 24, 2021
Accepted : January 24, 2022
* Corresponding Author : Yujin Lim(yujin91@sookmyung.ac.kr)

스 엣지 컴퓨팅(Multi-access Edge Computing, MEC)이 차세대 기술로 부상했다[3]. 엣지 컴퓨팅은 사용자(단말)와 지리적으로 근접하기 때문에 빠르게 연산 결과를 제공함으로써 서비스의 QoS 요구사항 만족도를 향상시킬 수 있다.

하지만, MECS(MEC Server)는 클라우드 서버에 비해 컴퓨팅 자원이 제한적이라 MECS가 혼잡해진다면 MECS에서의 대기 시간이 길어져 낮은 지연시간으로 서비스를 제공하기 어려울 수 있다. 그러나 MECS가 밀집하게 배치된 환경이라면 여러 MECS들의 협업을 통해 서비스의 총 지연시간을 더욱 단축시킬 수 있다. 즉, 하나의 태스크를 여러 개의 서브 태스크들로 분할하고, 여러 MECS들에 의해 서브 태스크들이 병렬적으로 수행됨으로써 서비스 지연시간을 단축시킬 수 있다. 이를 태스크 파티셔닝(partitioning)이라 한다.

기존 연구에서는 차량 또는 MECS에서 태스크를 분할하는 것에 관한 태스크 파티셔닝 기법들이 제안되었다. [4-6]에서는 차량에서 태스크를 파티셔닝하고, 서브 태스크들을 다른 MECS로 오프로딩하는 기법들이 제안되었다. [4]는 지연시간과 신뢰성을 절충하기 위해 서브 태스크들을 다른 MECS들에게 순차적으로 오프로딩하는 휴리스틱 기법을 제안하였고, [5]에서는 종속적인 속성을 가진 태스크들을 파티셔닝하고, 스케줄링을 통해 서브 태스크들의 오프로딩을 결정하는 기법을 제안하였다. [6]은 지연시간과 에너지 소비량을 최소화하기 위해 태스크 파티셔닝 및 전송 전력 전략에 따라 오프로딩하는 기법을 제안했다. 그리고 [7-9]에서는 태스크를 특정 MECS로 먼저 오프로딩하고, 해당 MECS에서 태스크를 파티셔닝하여 다른 MECS로 마이그레이션(migration)하는 기법들이 제안되었다. [7]은 시스템의 자원 효율성을 높이고 서비스 지연시간을 줄이기 위해, 파티셔닝하여 다른 MECS들로 마이그레이션하는 기법을, [8]에서는 지연시간과 에너지 소비를 최소화하기 위해 MECS로 오프로드하고 파티셔닝하는 기법을 제안하였다. [9]에서는 엣지 컴퓨팅과 클라우드 컴퓨팅이 협업하여 총 지연시간을 줄이기 위한 최적의 태스크 파티셔닝 전략을 제안하였다.

차량에서 파티셔닝하여 오프로딩하는 기법[4-6]은 차량과 MECS 사이의 무선 링크를 통해 서브 태스크들을 각각 오프로딩한다. 여기서, 무선 링크의 상태에 따라 전체 지연시간에 영향을 미칠 수 있는 반면에 MECS에서 분할하는 기법[7-9]은 통신 상태가 좋은 가장 가까운 MECS로 태스크를 오프로딩하고, 해당 MECS가 파티셔닝하여 고품질 백홀 링크를 통해 서브 태스크들을 다른 MECS들에게 마이그레이션한다. 그러나 MECS에서의 파티셔닝은 서브 태스크들을 최종 MECS로 전송하는데 두 홉이 필요하기 때문에 차량에서의 파티셔닝 기법보다 항상 좋은 것은 아니다[10].

본 논문에서는 MEC 시스템에서 차량에서 파티셔닝하여 오프로딩하는 기법과 MECS에서 파티셔닝하여 다른 MECS들로 마이그레이션하는 기법을 다양한 시나리오에 적용하여 최적의 파티셔닝 기법을 무엇인지 확인하고자 성능을 비교 분석하였다. 특히, 오프로딩과 마이그레이션 연구에서의 중요 요소인 대상 선정 그리고 파티셔닝 연구에서의 중요 요소인 개수 선정에 따라 서비스 지연시간, 서비스 거절률(blocking rate) 그리고 에너지 소비량 측면에서 성능을 비교 분석한다.

본 논문의 구성은 다음과 같다. 2장에서는 시스템 모델에 대해 기술하고, 3장에서는 실험 및 결과에 대해 분석하고, 마지막으로 4장에서는 결론 및 향후 연구를 다룬다.

## 2. 시스템 모델

본 논문에서 고려하는 MEC 시스템은 $M$개의 MECS가 제한된 환경에 밀집하게 배치되어 있으며, 서로 태스크를 공동으로 수행할 수 있는 협력적인 관계에 있다고 가정한다. 그리고 각 MECS는 FIFO 방식으로 작동하는 큐를 사용하며, MECS 사이는 백홀 링크를 통해 통신한다. 한 타임 슬롯에 차량 $n$은 $\{l_n, z_n, d_n^{\max}\}$으로 구성된 하나의 태스크를 발생시키며, $l_n$, $z_n$ 그리고 $d_n^{\max}$는 각각 입력 크기, 연산하기 위해 요구되는 CPU 사이클 그리고 태스크를 완료하기까지 최대 허용 가능한 지연시간이다. 태스크는 독립적인 관계에 있는 여러 개의 서브 태스크들로 구성돼있다고 가정하였다. $\gamma$는 파티셔닝 비율로 0과 1사이의 값을 가진다. 차량 $n$에서 발생한 태스크가 $I$개의 서브 태스크로 파티셔닝되었을 때, 서브 태스크 $i$가 MECS $m_i$으로 오프로드할 때, 전송 지연시간은 다음과 같다.

$$d_{n_i m_i}^{trans} = \frac{\gamma l_n}{r_{n,m_i}} \tag{1}$$

이때, $r_{n,m_i}$는 차량 $n$과 MECS $m_i$사이의 전송 속도로, 다음과 같다.

$$r_{n,m_i} = W \log_2 (1 + \frac{P_n h_{n,m_i}}{\sigma^2}) \tag{2}$$

$W$는 채널에 할당된 전송 대역폭, $P_n$는 차량의 송신 전력 그리고 $\sigma^2$는 잡음 전력이다. $h_{n,m_i}$는 차량 $n$과 MECS $m_i$사이의 채널 이득이다. 태스크의 연산 결과는 태스크의 입력 크기보다 훨씬 작아 다운 지연시간은 고려하지 않는다. 컴퓨팅 능력이 $c_{m_i}$인 MECS $m_i$에서 차량 $n$의 서브 태스크 $i$가 수행될 때의 연산 지연시간은 다음과 같다.

$$e_{n_i m_i} = \frac{\gamma z_n}{c_{m_i}} \tag{3}$$

만약 MECS에 의해 수행되고 있는 태스크가 없는 경우, 현재 태스크의 수행 시작 시간은 도착 시간이 된다. 그렇지 않으면 MECS 큐에서 대기하는 태스크들의 실행이 완료되어야 현재 태스크가 MECS에서 수행될 수 있다. $a_{n_i m_i}$는 차량 $n$의 서브 태

스크 $i$가 MECS $m_i$의 큐에 도착한 시간이며, $s_{n_k m_i}$와 $f_{n_k m_i}$는 각각 MECS $m_i$에서 차량 $n$의 서브 태스크 $i$가 수행을 시작한 시간과 종료된 시간을 나타낸다. 시작 시간은 큐에서 대기하는 마지막 태스크의 종료 시간에 따라 달라진다. 이때 큐에서 대기하는 마지막 태스크를 $j$라 하면, $s_{n_k m_i} = max\{f_{j,m_i},\ a_{n_k m_i}\}$으로 표현할 수 있다. 그리고 종료시간은 다음과 같다.

$$f_{n_k m_i} = s_{n_k m_i} + e_{n_k m_i} \tag{4}$$

차량 $n$의 서브 태스크 $i$가 MECS $m_i$에서 소요하는 지연시간은 다음과 같다.

$$t_{n_k m_i} = w_{n_k m_i} + e_{n_k m_i} \tag{5}$$

$w_{n_k m_i}$는 MECS $m_i$에서의 태스크 대기시간으로 태스크의 큐 도착 시간과 태스크 실행 시작 시간의 간격으로 다음과 같다. 만약 태스크가 즉시 수행되었다면 대기 시간이 없다.

$$w_{n_k m_i} = s_{n_k m_i} - a_{n_k m_i} \tag{6}$$

Fig. 1의 보라색처럼 차량에서 태스크를 파티셔닝하여 각각 다른 MECS들로 오프로딩하는 기법의 에너지 소비량은 다음과 같다. 태스크가 총 $I$개로 파티셔닝되었을 때, 각각 MECS들로 오프로딩하는 전송 지연시간의 합만큼 에너지를 소비한다.

$$E_n^{off} = P_n \sum_i^I d_{n_k m_i}^{trans} \tag{7}$$

$P_n$는 차량의 송신 전력이다. Fig. 1의 파란색의 경우는 차량의 전체 태스크를 차량과 가장 가까우면서 신호의 세기가 가장 센 MECS $m$로 오프로딩한다. 이때, 전송 지연시간은 (1)에서 $\gamma = 1$로, $d_{n,m}^{trans} = l_n / r_{n,m}$와 같다. 그다음 MECS $m$에서 태스크를 파티셔닝하여 다른 MECS들로 마이그레이션한다. 따라서 MECS에서 파티셔닝하여 마이그레이션하는 기법의 에너지 소비량은 다음과 같다.

$$E_n^{mig} = P_n d_{n,m}^{trans} \tag{8}$$

그리고 차량에서 파티셔닝하여 오프로딩하는 기법과 MECS에서 파티셔닝하여 마이그레이션하는 기법의 서비스 지연시간은 서브 태스크들 중 마지막 서브 태스크의 종료 시간에 첫 번째 서브 태스크의 도착 시간의 차이에 첫 번째 서브 태스크를 오프로딩하는 지연시간을 더하여 계산하였다. 이 2가지의 기법들을 본 논문에서는 서비스 지연시간과 거절률 그리고 차량이 소비하는 에너지량 측면에서 성능을 비교 분석한다. (1)과 (6)에서 무선 통신 상태와 MECS의 대기시간이 전체 지연시간에 영향을 미치며, (7)과 (8)에서는 오프로딩하는 전송 지연시간만큼 차량이 에너지를 소비한다는 것을 보여준다.

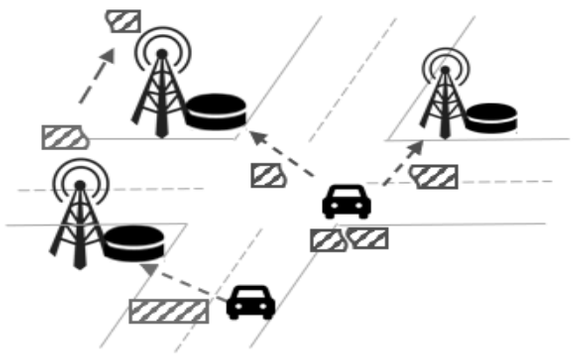


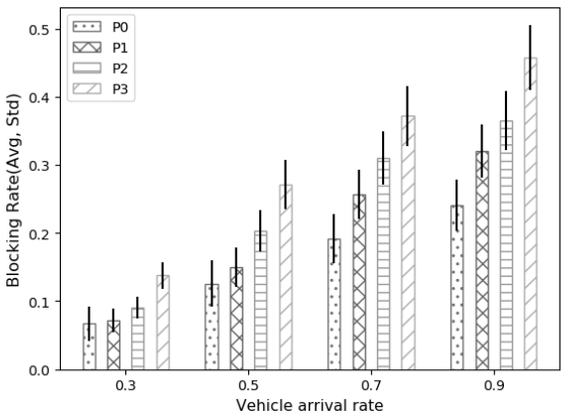
Fig. 1. Task Partitioning Strategy in MEC System

## 3. 실험 및 성능 비교 분석

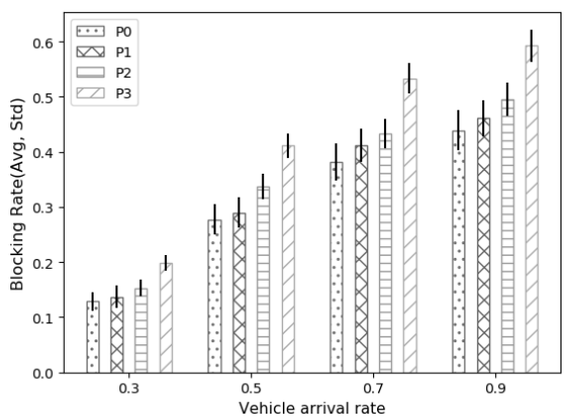
오프로딩을 이용한 차량에서의 파티셔닝 기법과 마이그레이션을 이용한 MECS에서의 파티셔닝 기법들의 성능 비교를 위한 실험을 진행하였다. 총 15개의 MECS가 밀집하게 배치된 환경을 가정하였고, 각 MECS의 컴퓨팅 능력은 10GHz이다. 차량은 푸아송 분포를 사용하여 발생하였으며, 이동속도는 약 10-30m/s이다. 태스크의 크기 $l_n$은 [2,6]Kbits, 요구되는 CPU 사이클 $z_n$는 [0.1,1]Gcycles, 그리고 최대 허용 지연시간 $d_n^{\max}$는 [0.1,0.2]s 범위 내에서 무작위로 발생하였다. 차량의 송신 전력 $P_n$는 600mW이고, 잡음 전력 $\sigma^2$은 $10^{-11}$mW이다. 그리고 차량과 MECS사이의 경로 손실은 $128.1 + 37.6\log_2(distance(km))$이다.

본 논문에서는 차량에서 파티셔닝을 하고 각 MECS들에게 오프로딩하는 기법(Partitioning+Offloading method)과 일단 차량이 MECS로 오프로딩한 후 해당 MECS에서 파티셔닝하여 다른 MECS들에게 마이그레이션하는 기법(Partitioning+Migration method)으로 나눠 실험을 진행하였다. 그리고 대상 MECS 선정 기법에 따른 성능 비교를 위해 거리를 고려하여 대상 MECS들을 선정하는 기법, MECS의 부하 상태를 고려하여 MECS들을 선정하는 기법 그리고 랜덤으로 대상 MECS들을 선정하는 기법으로 나눠 실험하였다. 또한, 몇 개의 서브 태스크로 파티셔닝하는지에 따른 성능을 비교하고자 총 4개의 기법을 사용하였다. 파티셔닝을 진행하지 않고 전체 태스크를 하나의 MECS에서 실행하는 *P0* 기법, 파티셔닝을 1번 진행하여 2개의 서브 태스크로 분할하여 실행하는 *P1* 기법, 파티셔닝을 2번 진행하는 *P2* 기법, 그리고 파티셔닝을 3번 진행하는 *P3* 기법으로 실험을 진행하였다.

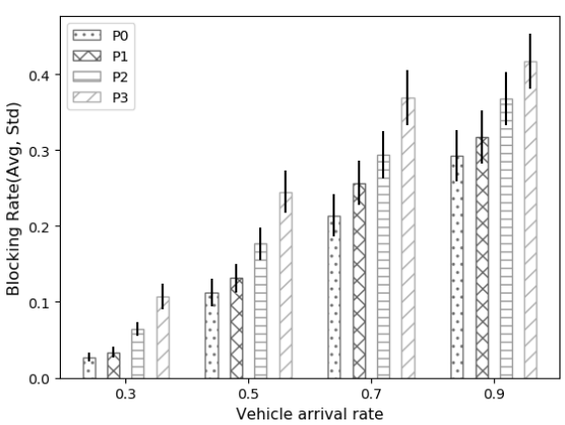
Fig. 2는 차량에서 파티셔닝하여 오프로딩하는 환경에서 서비스 거절률에 대한 차이를 보여주는 실험이다. 여기서, 서비스 거절률은 서버의 큐가 꽉 차서 더 이상 태스크를 수




(A) When the Nearest MECS is Selected as the Target Server




(B) When the Least-loaded MECS is Selected as the Target Server



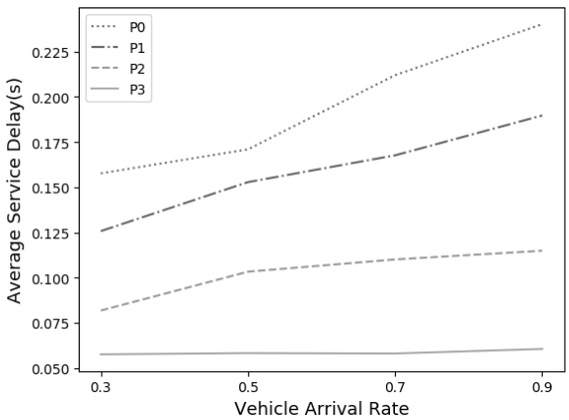

(C) When the MECS is Randomly Selected as the Target Server

Fig. 2. Comparison of the Blocking Rate using Partitioning+Offloading Method

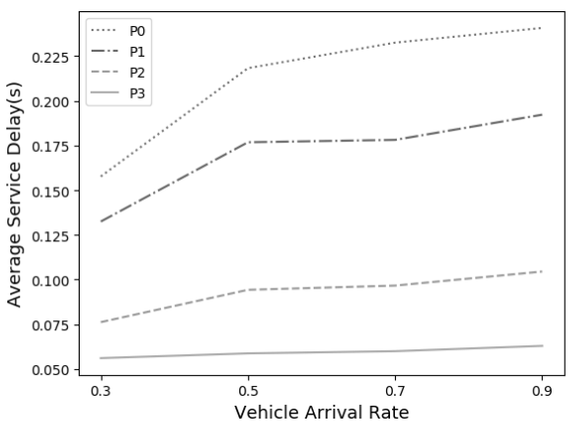
용할 수 없어 거절되는 수를 의미하며, 이를 전체 발생한 태스크 수로 나누어 정의하였다. 차량의 위치를 기반으로 Fig. 2(A)는 지리적으로 가장 가까운 MECS들로, Fig. 2(B)는 로드가 작은 MECS들로, 그리고 Fig. 2(C)는 랜덤으로 파티셔닝 대상 서버를 선정하여 오프로딩하였다. 파티셔닝 대상 선정 방법 간 성능을 비교하면, 로드가 작은 MECS들로 파티셔닝하여 오프로딩하는 Fig. 2(B)가 다른 Fig. 2(A), 2(C)와 비교하였을 때 약 36-40%만큼 성능이 낮다. 이는 차량과 MECS 간 거리나 랜덤으로 대상을 선정한 것보다 특정 MECS로 몰리는 경우가 많기 때문이다. 다시 말해서, 차량들은 도로 위에서 다양한 위치에 분포되어 있기 때문에 차량에서 가장 인접한 MECS가 차량에 따라 다를 수 있다. 그러나 작업 부하가 가장 낮은 MECS의 경우는 차량의 위치에 상관없이 그 순간에 단 하나이므로 모든 태스크 처리 요청이 집중되기 때문이다. 대상 MECS를 랜덤 선정한 Fig. 2(C)는 거리를 고려한 Fig. 2(A)보다 약 5% 나은 성능을 보인다. 이는 거리 기반 선정 기법의 경우 도로 위 차량의 위치가 얼마나 고르게 분포되어 있는지에 영향을 받는 것으로 해석할 수 있다. 파티셔닝 개수 측면에서, 파티셔닝을 진행하지 않는 *P0* 기법이 파티셔닝을 진행한 다른 기법 *P1, P2, P3*보다 더 나은 성능을 보인다. 그리고 파티셔닝을 많이 진행할수록 성능이 안 좋아지는데 이는 MECS에서 수행해야 할 태스크의 수가 많아져 MECS가 모두 수용하지 못해 거절되는 태스크의 수가 증가하기 때문이다. 이러한 현상은 차량의 도착률이 높아질수록 성능이 안 좋아지는 것에서도 확인할 수 있다. 낮은 로드를 가진 MECS를 선정하는 Fig. 2(B)는 다른 환경에 비해 파티셔닝 개수에 따른 거절률의 성능 차이가 약 5-18%로 작은데, 이는 특정 MECS로 항상 몰리기 때문에 파티셔닝의 유무나 개수에 상관없이 거절률이 높아 기법별 차이가 크지 않다.

Fig. 3은 차량에서 파티셔닝하여 각 MECS들에게 오프로딩하는 환경에서 서비스 지연시간 차이를 비교하였다. 파티셔닝 대상 선정 기법별로 성능을 비교한 결과, 파티셔닝 대상을 랜덤 선정한 Fig. 3(C), 거리로 선정한 Fig. 3(A), 그리고 MECS 부하 상태를 고려한 Fig. 3(C) 순으로 성능이 좋다. 랜덤 선정의 성능이 좋은 이유는 협력적인 관계에 있는 MECS들 중 랜덤으로 선정하기 때문에 다른 기법들에 비해 여러 MECS들을 폭넓게 선정할 수 있고 이로 인하여 서비스 거절률이 낮기 때문이다. 서비스 거절률이 높다는 것은 특정 MECS로 몰린다는 의미로, 이는 MECS 내 대기시간이 길어진다는 것이고, 이는 곧 서비스 지연시간이 증가하는 것이다. 그래서 서비스 거절률이 높게 나온 거리나 부하 기반 MECS 대상 선정 기법들을 사용한 환경에서의 성능이 안 좋다. 파티셔닝 개수에 대한 성능 차이를 보면, *P0* 기법이 *P1, P2, P3* 기법들보다 지연시간이 더 소요된다. 여러 MECS에서 공동으로 태스크를 수행하기 때문에 연산에 필요한 지연시간이 단축되므로 파티셔닝을 여러 번 진행할수록 나은 성능을 보인다.

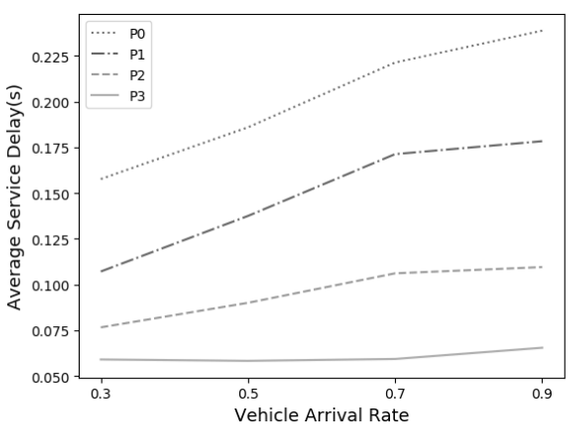
Fig. 4는 차량에서 가장 가까운 MECS로 오프로딩을 하고, MECS에서 파티셔닝하여 다른 MECS들에게 마이그레이션하는 환경에서 서비스 거절률에 대한 성능 차이를 보여준다. Fig. 4(A)는 오프로딩한 MECS에서 지리적으로 가까운 MECS들, Fig. 4(B)와 4(C)는 각각 로드가 작은 MECS들과 랜덤하게 선정된 MECS들을 파티셔닝 대상을 선정하여 마이




(A) When the Nearest MECS is Selected as the Target Server

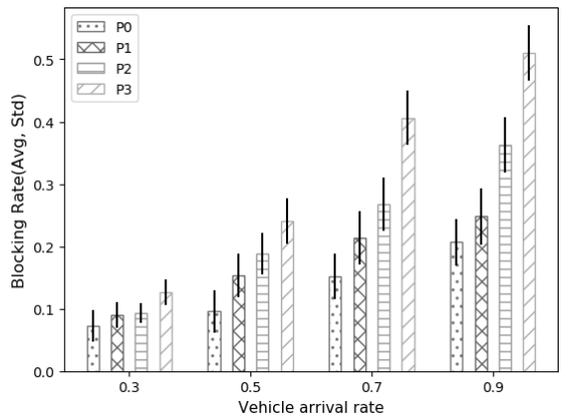



(B) When the Least-loaded MECS is Selected as the Target Server

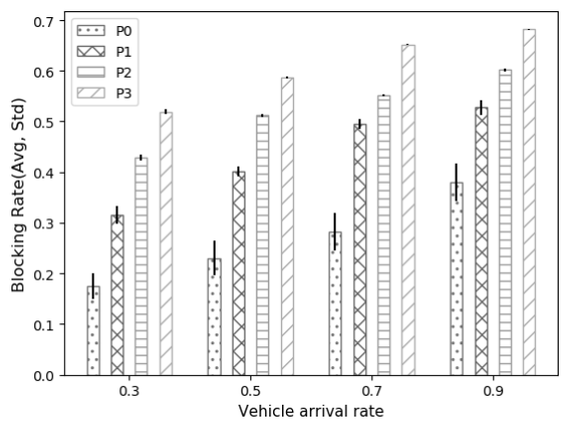



(C) When the MECS is Randomly Selected as the Target Server

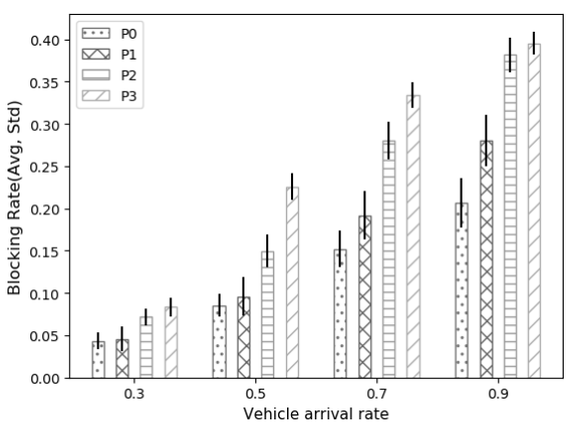
Fig. 3. Comparison of the Average Service Delay using Partitioning+Offloading Method




(A) When the Nearest MECS is Selected as the Target Server



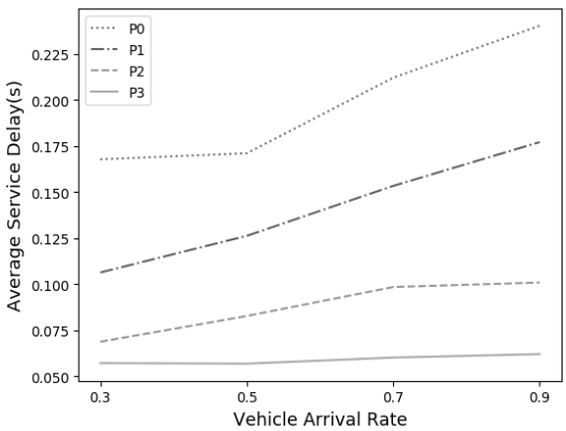

(B) When the Least-loaded MECS is Selected as the Target Server



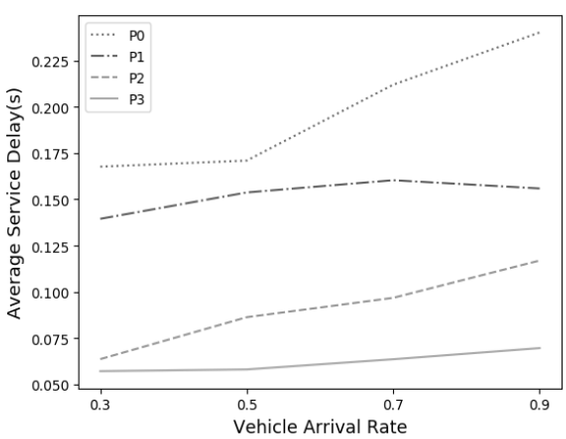

(C) When the MECS is Randomly Selected as the Target Server

Fig. 4. Comparison of the Blocking Rate using Partitioning+Migration Method

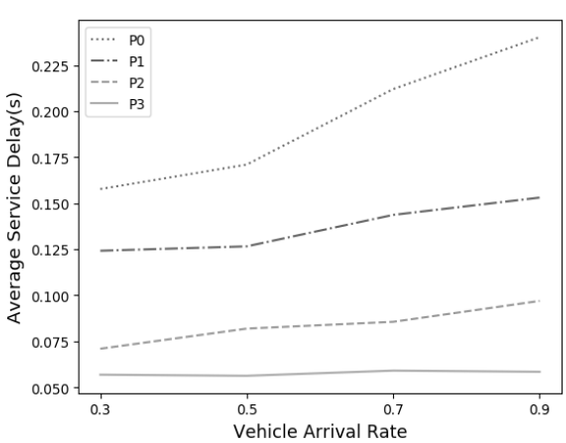
그레이션한다. 파티셔닝 대상 선정 측면에서 성능을 비교한 결과, 로드가 작은 MECS들로 마이그레이션하는 Fig. 4(B)가 다른 기법들에 비해 약 53-58%만큼 높은 거절률을 보여줬다. 이는 협력관계에 있는 MECS들 중 로드가 작은 MECS들을 선정하는 것이라, 다른 MECS들로부터의 오프로딩이 로드가 작은 MECS를 대상으로 한꺼번에 몰리기 때문이다. 그리고 거리와 랜덤으로 대상을 선정한 환경은 약 11% 정도 성능 차이가 난다. 협력적인 관계에 있는 MECS들 중 랜덤으로 선정하는 것이라 거리 기반으로 선택하는 것보다 폭넓게 선택할 수 있기 때문이다. *P0* 기법이 파티셔닝을 진행한 다른 기법들보다 약 39-45%만큼 성능이 좋다. 파티셔닝을 많이 진행한 *P3* 기법의 성능이 가장 안 좋다.




(A) When the Nearest MECS is Selected as the Target Server




(B) When the Least-loaded MECS is Selected as the Target Server




(C) When the MECS is Randomly Selected as the Target Server

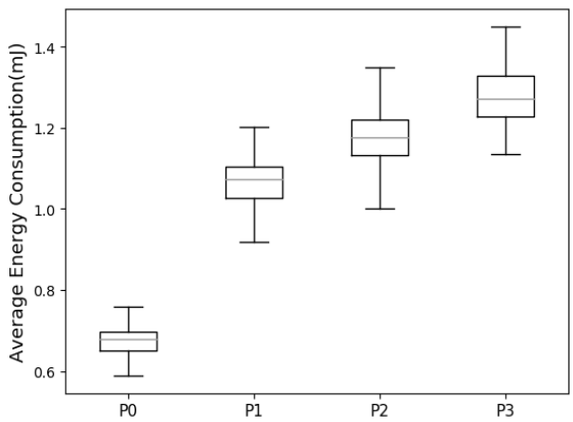
Fig. 5. Comparison of the Average Service Delay using Partitioning+Migration Method

그리고 거리로 파티셔닝 대상을 선정한 Fig. 2(A)와 Fig. 4(A)를 비교해 보면, 마이그레이션을 이용하여 파티셔닝한 Fig. 5(A)가 약 5% 정도 성능이 좋다. 이는 오프로딩을 이용한 기법이 파티셔닝한 태스크의 개수만큼 차량에서 MECS들로 여러 번 오프로딩하므로 마이그레이션 이용 기법보다 전송 지연시간이 더 소요되어 나타나는 차이이다. MECS의 부하 상태를 고려하여 오프로딩을 이용한 Fig. 2(B)와 마이그레이션 이용한 Fig. 4(B)를 비교해 보면, 오프로딩 이용 기법이 마이그레이션 이용 기법보다 약 22% 더 나은 성능을 보인다. 차량과 통신할 수 있는 MECS들 중 로드가 낮은 MECS들을 선정한 오프로딩 기법과 달리, 마이그레이션 이용 기법은 협력관계에 있는 모든 MECS들 중 로드가 작은 MECS들을 선정하므로 특정 MECS로 몰리는 경향이 더 심화되기 때문이다. 그리고 랜덤으로 대상을 선정한 Fig. 2(C)와 4(C)는 마이그레이션 이용 기법이 오프로딩 이용 기법보다 약 11% 좋다. 따라서 결론적으로, MECS 상태를 고려하면서 특정 MECS로 쏠리지 않도록 대상을 선정하는 것이 중요하다고 할 수 있다.

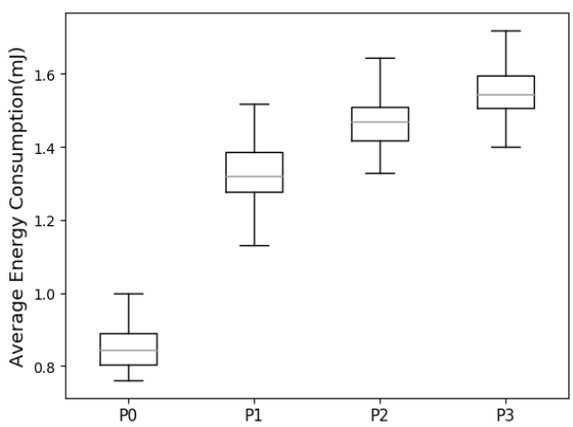
Fig. 5는 MECS에서 파티셔닝을 하고 마이그레이션을 이용한 환경에서 서비스 지연시간에 대한 성능을 비교하였다. 파티셔닝 대상 선정 측면에서 Fig. 5를 비교한 결과, 파티셔닝 대상을 랜덤으로 선정한 Fig. 5(C), 거리로 선정한 Fig. 5(A), 그리고 MECS 부하 상태로 선정한 Fig. 5(B) 순으로 성능이 안 좋다. 파티셔닝 개수에 대한 성능 차이를 보면, *P0* 기법이 파티셔닝을 진행한 *P1, P2, P3* 기법들보다 지연시간이 더 소요되며, 여러 개의 서브 태스크로 파티셔닝할수록 지연시간이 단축된다. 여러 MECS에서 공동으로 수행하여 연산에 대한 부담이 줄기 때문이다.

지연시간 측면에서 오프로딩을 이용한 Fig. 3과 마이그레이션을 이용한 Fig. 5를 비교해 보면, 오프로딩 이용한 기법이 약 5-6% 정도 차이 난다. 이는 파티셔닝 대상이나 개수에 따라 발생한 차이가 아니라 차량이 태스크를 오프로딩할 때의 전송 지연시간에서 발생한 차이이다. 서비스 지연시간 측면에서는 파티셔닝을 진행하는 것이 진행하지 않는 것보다 좋으며, 태스크를 여러 개의 서브 태스크로 파티셔닝하는 것이 지연시간을 단축시킬 수 있다. 하지만 이와 반대로 서비스 거절률 측면에서는 파티셔닝을 진행하지 않는 것의 성능이 좋으며, 서브 태스크들의 개수가 많아질수록 성능이 안 좋아진다.

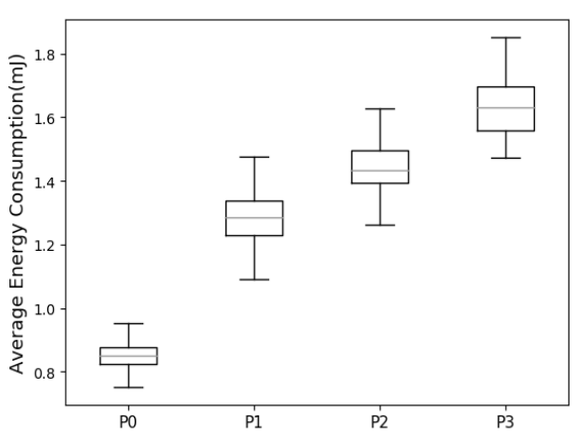
Fig. 6은 차량이 파티셔닝하여 오프로딩하는 환경에서 파티셔닝 개수에 따라 차량이 소비한 에너지량을 비교한 실험이다. 차량은 MECS로 태스크를 오프로딩할 때 에너지를 소비한다. 파티셔닝 대상 측면에서 Fig. 6을 비교해 보면, 거리를 고려하여 오프로딩을 한 Fig. 6(A)가 거리를 고려하지 않은 다른 환경들에서 보다 에너지 소비량이 적다. 그 이유는 전송 지연시간은 거리와 태스크의 크기가 좌우하는데, 특히, 차량과 MECS 간의 거리가 더 큰 영향을 미치기 때문이다. 그리고 파티셔닝 개수에 따른 에너지 소비량을 비교해 보면, 파티셔닝 진행하지 않는 *P0* 기법과 진행하는 *P1, P2, P3* 기법의 에너지 소비량 측면에서의 성능 차이는 약 36-45% 차이로 파티셔닝 진행 여부가 에너지 소비량에서 가장 크게 좌우하고, 파티셔닝을 진행한다면 파티셔닝을 많이 진행할수록 에너지 소비량이 커진다. 파티셔닝의 개수만큼 차량에서 MECS로 오프로딩을 하기 때문에 파티셔닝 개수가 많을수록




(A) When the Nearest MECS is Selected as the Target Server




(B) When the Least-loaded MECS is Selected as the Target Server




(C) When the MECS is Randomly Selected as the Target Server

Fig. 6. Comparison of the Average Energy Consumption using Partitioning+Offloading Method

에너지 소비량이 증가할 수밖에 없다. 그리고 마이그레이션을 이용한 파티셔닝 기법에서는 차량과 가장 가까운 MECS로 오프로딩하고 파티셔닝을 하는 기법이므로 파티셔닝 개수에 따른 에너지 소비량이 다르지 않다. 마이그레이션을 이용한 파티셔닝의 에너지 소비량은 Fig. 6(A)에서 파티셔닝을 진행하지 않은 *P0* 기법과 비슷하다.

서비스 지연시간 측면에서는 파티셔닝을 진행하는 것이 진행하지 않는 것보다 좋으며, 태스크를 여러 개의 서브 태스크로 파티셔닝하는 것이 지연시간을 단축하는 것이다. 하지만, 오프로딩을 이용한 기법에서는 그만큼 차량이 소비하는 에너지량이 증가한다.

## 4. 결 론

태스크를 파티셔닝하여 여러 MECS들의 협력을 통해 연산하는 것이 하나의 MECS에서만 수행할 때 보다 연산에 대한 부담이 적어 전체적인 지연시간을 단축시킬 수 있다. 본 논문에서는 오프로딩과 마이그레이션을 이용한 파티셔닝 기법들을 파티셔닝 대상 선정과 개수 변화에 따른 서비스 지연시간과 거절률, 그리고 에너지 소비량에 대한 성능을 비교 분석하였다. 파티셔닝 대상 선정에 따라 마이그레이션과 오프로딩 이용 기법의 성능이 달라짐을 확인하였다. 여러 개의 서브 태스크로 파티셔닝할수록 지연시간 측면에서의 성능은 향상되지만, 반대로 서비스 거절률 측면에서의 성능은 감소하게 된다. 또한, 차량의 에너지 소비량 측면에서도 파티셔닝 개수가 증가할수록 오프로딩 횟수가 증가하기 때문에 성능이 감소하게 됨을 확인하였다. 향후 연구에서는 보다 다양한 환경과 시나리오에서 실험을 진행하고, 태스크를 파티셔닝하는 방법에 대한 연구를 진행하고자 한다.

## References

[1] S. Raza, S. Wang, M. Ahmed, and M. R. Anwar, "A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions," *Wireless Communications and Mobile Computing*, Vol.2019, pp.1-19, 2019.

[2] 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, 2018.

[3] 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.

[4] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, Vol.6, pp.2169-3536, 2018.

[5] J. Liu and Q. Zhang, "Code-partitioning offloading schemes in mobile edge computing for augmented reality," *IEEE Access*, Vol.7, pp.11222-11236, 2019.

[6] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Transactions on Cognitive Communications and Networking*, Vol.6, No.4, pp.1122-1135, 2020.

[7] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, Vol.26, No.4, pp.1619-1632, 2018.

[8] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, Vol.64, No.10, pp.4268-4282, 2016.

[9] M. Feng, M. Krunz, and W. Zhang, "Joint task partitioning and user association for latency minimization in mobile edge computing networks," IEEE Transactions on Vehicular Technology, Vol.70, No.8, pp.8108-8121, 2021.

[10] J. Liu and Q. Zhang, "Adaptive task partitioning at local device or remote edge server for offloading in MEC," *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, pp.1-6, May, 2020.

**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