MEC 환경에서 오프로딩과 마이그레이션을 이용한 태스크 파티셔닝 기법의 성능비교

문성원, 구설원, 임유진
숙명여자대학교 IT 공학과
sungwon268@sookmyung.ac.kr, tjrgkr501@sookmyung.ac.kr, yujin91@sookmyung.ac.kr

Performance Comparison of Task Partitioning with Offloading and Migration in MEC

Sungwon Moon, Seolwon Koo, Yujin Lim
Dept. of IT Engineering, Sookmyung Women's University

요 약

5G 의 발전과 함께 차량과 IT 통신 기술을 융합한 어플리케이션들이 급증하면서 멀티 액세스 엣지 컴퓨팅(MEC)이 차세대 기술로 등장했다. 낮은 지연시간 안에 계산 집약적인 서비스들을 제공하기 위해 단독적인 MECS 서버(MECS)에서의 수행이 아닌 다수의 MECS 에서 동시에 연산을 수행할 수 있도록 태스크를 파티셔닝하는 기법이 주목받고 있다. 특히 차량이 다수의 MECS 로 태스크를 파티셔닝하여 오프로딩하는 기법과 하나의 MECS 로 오프로딩한 후 다른 MECS 들로 파티셔닝하여 마이그레이션하는 기법들이 연구되고 있다. 본 논문에서는 오프로딩과 마이그레이션을 이용한 파티셔닝 기법들을 서비스 지연시간과 차량의 에너지 소비량 측면에서 성능을 비교 분석을 하였다.

1. 서론

5 세대 통신 기술(5G)의 발전과 함께 사물인터넷(IoT) 시대에는 차량과 IT 통신 기술이 융합된 차량 통신 네트워크 기술과 이 기술을 활용한 자율주행, 차량 안전 서비스, 경로 계획 등 다양한 어플리케이션들이 연구되고 있다[1]. 이런 어플리케이션은 지연시간에 민감하며 계산 집약적이다. 이를 제한적인 컴퓨팅 자원을 갖는 차량에서 제공하기에는 한계가 있다. 그래서 낮은 지연시간 안에 계산 집약적인 서비스를 제공하기 위해 멀티 액세스 엣지 컴퓨팅(Multi-access Edge Computing, MEC)이 차세대 기술로 등장했다[2]. 기존 클라우드 컴퓨팅(cloud computing)과 달리, MEC 는 차량과 근접한 곳에서 데이터를 처리할 수 있어 연산 결과를 빠르게 제공받을 수 있다.

차량은 발생한 태스크를 MECS(MEC Server)로 오프로드(offload) 한다. 하지만, MECS 는 클라우드 서버에 비해 컴퓨팅 자원이 제한적이기 때문에 태스크들이 많아질수록 MECS 에서의 대기 시간이 길어져 제한된 지연시간 내에 서비스를 제공하기 어렵다. 그래서 태스크를 하나의 MECS 에서 실행시키는 것이 아닌 다수의 MECS 에서 동시에 수행시켜 서비스 지연시간을 단축시킬 수 있다. 즉, 태스크를 여러 개의 서브 태스크로 나눠 다수의 MECS 에서 수행할 수 있도록 태스크를 파티셔닝(partitioning)하는 것이다.

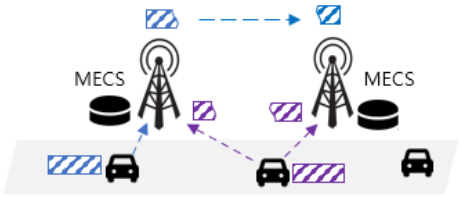
기존 연구에서는 차량에서 여러 개의 MECS 로 서브 태스크들을 오프로드하는 파티셔닝 기법[3-4]이 제안되었다. [3]은 서비스 지연시간을 최소화하기 위해 MECS 와 cloud 로, [4]는 서비스 지연시간과 차량의 에너지 소비량을 최소화하기 위해 차량에서 다수의 MECS 로 파티셔닝하여 오프로딩하는 기법을 제안하였다. 오프로딩을 이용한 파티셔닝 기법들 경우, 차량과 MECS 사이의 무선 링크들을 통해 오프로딩을 하기 때문에 무선 링크의 상태가 전체적인 지연시간에 영향을 미친다.

차량에서 하나의 MECS 로 태스크를 오프로드한 후 다른 MECS 들로 마이그레이션하는 파티셔닝 기법[5-6]도 있다. 서비스 지연시간을 줄이고 서비스 신뢰성 향상을 위해 차량과 가장 가까운 MECS 와 함께 수행할 MECS 를 선정하고 파티셔닝하여 마이그레이션하는 기법[5]과 시스템의 자원 효율성을 높이고 서비스 지연시간을 줄이기 위해 파티셔닝하여 다른 MECS 들로 마이그레이션하는 기법[6]을 제안하

였다. 마이그레이션을 이용한 파티셔닝 기법들이 다른 MECS 들로 마이그레이션할 때 무선 통신을 사용하기 때문에 이는 서비스 지연시간에 영향을 미친다.

그래서 본 논문에서는 오프로딩을 이용한 파티셔닝 기법들과 마이그레이션을 이용한 파티셔닝 기법들을 서비스 지연시간과 차량의 에너지 소비량 측면에서 성능을 비교 분석한다.

2. 시스템 모델



(그림 1) MEC 환경에서의 파티셔닝 기법들

본 논문에서 제안된 MEC 시스템은 M 개의 MECS 로 구성되어 있으며, 각 MECS는 RSU에 배치된 환경을 가정한다. 그리고 2 개의 MECS 가 공동으로 태스크를 수행할 수 있는 협력적인 관계에 있고 MECS 사이는 무선 통신을 한다고 가정한다. 차량의 위치와 가장 가까운 즉, 신호의 세기가 가장 센 서빙 MECS 를 MECS m 로, 이와 공동으로 태스크를 수행할 MECS를 MECS m'라 정의한다. 각각의 MECS 컴퓨팅 능력을 C_m, $C_{m'}$로 정의한다.

차량 n에서 발생한 태스크는 $\{S_n(t), Z_n(t), T_n^{max}\}$으로 정의되며, $S_n(t)$ 는 타임 슬롯 t 에 발생한 태스크의 크기이고, $Z_n(t)$는 태스크를 연산하기 위해 요구되는 CPU 사이클이다. 그리고 T_n^{max}는 태스크를 완료하기까지 허용 가능한 최대 지연시간이다. 태스크는 독립적인 관계에 있는 여러 개의 서브 태스크들로 구성되어 있다고 가정한다. γ는 파티셔닝 비율로 0 과 1 사이의 값을 가진다. MECS m, MECS m'에서 $\gamma Z_n(t)$, $(1-\gamma)Z_n(t)$ CPU 사이클만큼 태스크를 수행한다.

차량에서 발생한 태스크의 서브 태스크들을 MECS m와 MECS m'로 각각 오프로딩할 때의 전송 지연시간은 다음과 같다.

$$\begin{cases} \frac{\gamma S_n(t)}{R_{n,m}}, & if \quad offload \ to \ MECS \ m \\ \frac{(1-\gamma)S_n(t)}{R_{n,m'}}, & if \quad offload \ to \ MECS \ m' \end{cases} \quad (1)$$

이때, 차량 n과 MECS m사이의 전송 속도인 $R_{n,m}$은 다음과 같다.

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

W는 채널에 할당된 전송 대역폭, P_n는 차량의 송신 전력, σ^2는 잡음 전력이다. $H_{n,m}$는 차량과 MECS 사이의 채널 이득이다. 연산 결과는 태스크의 사이즈보다 작아 다운 지연시간은 고려하지 않는다. MECS m과 MECS m'에서 연산 지연시간은 다음과 같다.

$$\begin{cases} \frac{\gamma Z_n(t)}{C_m}, & if \ excute \ at \ MECS \ m \\ \frac{(1-\gamma) \ Z_n(t)}{C_{m'}}, & if \ excute \ at \ MECS \ m' \end{cases} \quad (3)$$

MECS m에서의 큐잉 지연시간은 다음과 같다.

$$T_m^Q(t) = \max_{t' \in \{0,1,\dots,t-1\}} \{ l_m^{comp}(t') - t \} \quad (4)$$

$l_m^{comp}(t)$는 t 에 발생한 태스크가 완료될 시간을 의미하며, 다음과 같다.

$$l_m^{comp}(t) = \mathrm{t} + \frac{\gamma \ Z_n(t)}{C_m} + T_m^Q(t) - 1 \quad (5)$$

그림 1의 보라색처럼 차량에서 발생한 태스크를 파티셔닝하여 2 개의 MECS 로 오프로딩하는 기법의 서비스 지연시간과 에너지 소비량은 다음과 같다.

$$T_n^O(t) = \frac{\gamma S_n(t)}{R_{n,m}} + \max\{\frac{\gamma \ Z_n(t)}{C_m} + T_m^Q(t), \frac{(1-\gamma)S_n(t)}{R_{n,m'}} + \frac{(1-\gamma) \ Z_n(t)}{C_{m'}} + T_{m'}^Q(t)\} \quad (6)$$

$$E_n^O(t) = P_n(\frac{\gamma S_n(t)}{R_{n,m}} + \frac{(1-\gamma)S_n(t)}{R_{n,m'}}) \quad (7)$$

차량에서 발생한 태스크를 MECS m로 오프로딩을 할 때의 전송 지연시간은 (1)에서 $\gamma = 1$로, $\frac{S_n(t)}{R_{n,m}}$ 과 같다. 그리고 MECS m에서 MECS m'로 마이그레이션하는 전송 지연시간은 $\frac{(1-\gamma)S_n(t)}{R_{m,m'}}$과 같다. 그림 1 의 파란색처럼 차량의 태스크를 MECS m 로 오프로드하고 MECS m'로 마이그레이션하는 파티셔닝 기법의 서비스 지연시간과 에너지 소비량은 다음과 같다.

$$T_n^M(t) = \frac{S_n(t)}{R_{n,m}} + \max\{\frac{\gamma \ Z_n(t)}{C_m} + T_m^Q(t), \frac{(1-\gamma)S_n(t)}{R_{m,m'}} + \frac{(1-\gamma) \ Z_n(t)}{C_{m'}} + T_{m'}^Q(t)\} \quad (8)$$

$$E_n^M(t) = P_n \frac{S_n(t)}{R_{m,m'}} \quad (9)$$

본 논문에서는 오프로딩과 마이그레이션을 이용한 파티셔닝 기법들을 서비스 지연시간과 차량의 에너지 소비량에 대해 성능 비교를 한다. (6)과 (8)에서 무선 통신 상태와 MECS 의 큐잉 지연시간이 총 지연시간에 영향을 미치고, (7)과 (9)에서는 오프로딩하는 통신 시간이 에너지 소비량에 영향을 미침을 보여준다.

3. 태스크 파티셔닝 기법

본 논문에서는 차량에서 2 개의 MECS 로 파티셔닝하여 오프로딩하는 기법과 태스크를 하나의 MECS 로 오프로드하고 다른 MECS 로 파티셔닝하여 마이그레이션하는 기법을 서비스 지연시간과 차량의 에너지 소비량 측면에서 성능 비교한다. 협력적인 관계에 있는 MECS m과 MECS m'에서 태스크를 공동으로 연산한다. 최적의 파티셔닝은 2 개의 MECS 에서 동시에 연산이 완료될 때이며, 이때 지연시간을 최소화할 수 있는 최적의 파티셔닝 비율 γ을 계산할 수 있다.

오프로딩을 이용한 파티셔닝 기법에서 2 개의 MECS 에서 연산이 수행될 때는 MECS m에서의 큐잉 지연시간과 연산 지연시간이 MECS m'로의 전송 지연시간, 큐잉 지연시간 그리고 연산 지연시간의 합이

같은 경우로 다음과 같다.

$$\frac{\gamma\ Z_n(t)}{C_m} + T_m^Q(t) = \frac{(1-\gamma)S_n(t)}{R_{n,m'}} + \frac{(1-\gamma)\ Z_n(t)}{C_{m'}} + T_{m'}^Q(t) \quad (10)$$

(10)를 정리하여 (6)에 적용하면 최적의 오프로딩 파티셔닝 비율이 다음과 같이 계산된다.

$$\gamma_n^{O*} = \frac{\frac{S_n(t)}{R_{n,m'}} + \frac{Z_n(t)}{C_{m'}} + T_{m'}^Q(t) - T_m^Q(t)}{\frac{S_n(t)}{R_{n,m'}} + \frac{Z_n(t)}{C_{m'}} + \frac{Z_n(t)}{C_m}} \quad (11)$$

이때, MECS m'의 큐잉 지연시간이 MECS m의 연산 지연시간과 큐잉 지연시간의 합보다 크거나 같다면 파티셔닝을 하지 않고 MECS m에서 태스크를 수행한다. 이와 달리, MECS m 의 큐잉 지연시간이 MECS m'의 연산 지연시간과 큐잉 지연시간 그리고 MECS m에서 MECS m'로 전송되는 지연시간의 합보다 크거나 같다면 MECS m'에서 태스크를 수행한다. 이를 다음과 같이 표현할 수 있다.

$$\begin{cases} \gamma_n^{O*} = 1,\ if\ \ T_{m'}^Q(t) \geq \frac{\gamma\ Z_n(t)}{C_m} + T_m^Q(t) \\ \gamma_n^{O*} = 0,\ if\ \ T_m^Q(t) \geq \frac{(1-\gamma)S_n(t)}{R_{m,m'}} + \frac{(1-\gamma)\ Z_n(t)}{C_{m'}} + T_{m'}^Q(t) \end{cases} \quad (12)$$

마이그레이션을 이용한 파티셔닝 기법에서 2 개의 MECS 에서 동시에 연산이 완료되기 위해서 MECS m 의 큐잉 지연시간과 연산 지연시간이 MECS m'로 마이그레이션하는 지연시간, MECS m'의 연산 지연시간과 큐잉 지연시간의 합이 같은 경우로 다음과 같다.

$$\frac{\gamma\ Z_n(t)}{C_m} + T_m^Q(t) = \frac{(1-\gamma)S_n(t)}{R_{m,m'}} + \frac{(1-\gamma)\ Z_n(t)}{C_{m'}} + T_{m'}^Q(t) \quad (13)$$

그리고 (13)를 정리하여 (8)에 적용하면 최적의 마이그레이션 파티셔닝 비율은 다음과 같이 계산된다.

$$\gamma_n^{M*} = \frac{\frac{S_n(t)}{R_{m,m'}} + \frac{Z_n(t)}{C_{m'}} + T_{m'}^Q(t) - T_m^Q(t)}{\frac{S_n(t)}{R_{m,m'}} + \frac{Z_n(t)}{C_{m'}} + \frac{Z_n(t)}{C_m}} \quad (14)$$

(12)와 마찬가지로 다음과 같은 조건을 만족하면 태스크를 파티셔닝을 하지 않고 MECS m 또는 MECS m'에서 태스크를 수행한다.

$$\begin{cases} \gamma_n^{M*} = 1,\ if\ \ T_{m'}^Q(t) \geq \frac{\gamma\ Z_n(t)}{C_m} + T_m^Q(t) \\ \gamma_n^{M*} = 0,\ if\ \ T_m^Q(t) \geq \frac{(1-\gamma)S_n(t)}{R_{m,m'}} + \frac{(1-\gamma)\ Z_n(t)}{C_{m'}} + T_{m'}^Q(t) \end{cases} \quad (15)$$

(11)과 (14)를 각각 (6)과 (8)에 적용하면, 오프로딩과 마이그레이션을 이용한 파티셔닝 기법에서의 서비스 지연시간은 다음과 같다.

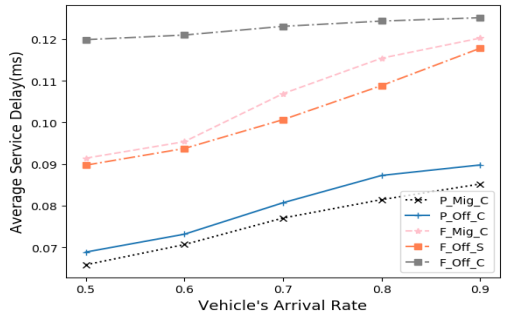
$$T_n^{O*} = \left(\frac{S_n(t)}{R_{n,m}} + \frac{Z_n(t)}{C_m}\right)\gamma_n^{O*} + T_m^Q \quad (16)$$

$$T_n^{M*} = \left(\frac{S_n(t)}{R_{n,m}} + \frac{Z_n(t)}{C_m}\gamma_n^{M*}\right) + T_m^Q \quad (17)$$

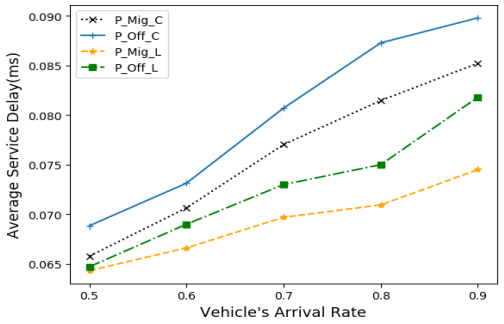
4. 실험 및 성능 비교 분석

파티셔닝 기법들의 성능 비교를 위한 실험을 진행하였다. 총 10 개의 MECS 가 배치된 환경이며, MECS 의 반경은 $250m$이고, 컴퓨팅 능력은 10 GHz이다. 푸아송 분포를 사용하여 차량을 발생시켰으며, 태스크의 크기 S_n, 요구되는 CPU 사이클 Z_n 그리고 최대 허용 지연시간 T_n^{max}는 [2,6] $Kbits$, [0.2,6] $Gcycles/Kbits$ 그리고 [100, 200]ms 범위 내에서 무작위로 발생하였다. 차량과 MECS 사이의 경로 손실은 $140.7 + 36.7\log_{10}(distance(km))$이고, 차량의 송신 전력 P_n은 $600mW$이다.

본 논문에서 전체(full) 오프로딩, 전체 마이그레이션이 아닌 파티셔닝 기법을 고려한 이유를 보여주기 위해 전체 기법들과 파티셔닝 기법들의 비교 실험을 진행하였다. 파티셔닝 기법인 *P_Mig_C, P_Off_C* 와 전체 기법인 *F_Mig_C, F_Off_S, F_Off_C* 를 비교하였다. *P_Mig_C* 는 서빙 MECS 에서 이웃 MECS 들 중 가장 가까운 MECS 로 파티셔닝하여 마이그레이션하는 기법이다. *P_Off_C* 는 서빙 MECS 와 그다음으로 가까운 MECS 로 파티셔닝하여 오프로딩하는 기법이다. *F_Mig_C* 는 서빙 MECS 에서 그다음으로 가장 가까운 MECS 로 전체 마이그레이션하는 기법이다. *F_Off_S* 는 서빙 MECS 로, *F_Off_C* 는 서빙 MECS 외에 가장 가까운 MECS 로 전체 오프로딩하는 기법이다.



(그림 2) 서비스 지연시간 비교



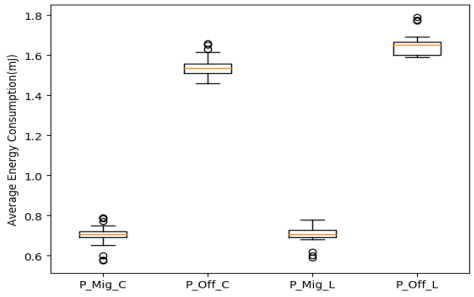
(그림 3) 파티셔닝 기법들간 서비스 지연시간 비교

그림 2 에서 오프로딩을 이용한 파티셔닝 기법인 *P_Off_C* 는 전체 오프로딩 기법인 *F_Off_S* 과 *F_Off_C* 보다 약 22%, 34.8%만큼 낮은 지연시간으로 서비스를 제공한다. *F_Off_S* 과 *F_Off_C* 의 성능 차이는 차량과 MECS 거리적인 요소로 인한 통신 상태 차이 때문이다. 마이그레이션을 이용한 파티셔닝 기법인 *P_Mig_C* 는 전체 마이그레이션 기법인 *F_Mig_C* 보다 약 28.2% 그리고 마이그레이션하지 않고 서빙 MECS 에서 수행하는 *F_Off_S* 보다 약 25.6% 만큼 성능 차이가 난다. 이를 통해 태스크를 단독적으로 수행하는 것보다는 파티셔닝하여 다수의

MECS 에서 수행하는 것이 별도의 전송 지연시간이 발생하여도 연산에 대한 부담을 덜어줘 지연시간을 최소화할 수 있음을 확인하였다.

그림 3 는 파티셔닝 기법들을 서비스 지연시간 측면에서 비교했을 뿐만 아니라 어떤 MECS 와 협업하는지에 따라 성능 차이가 존재하는지 확인하기 위해 진행한 실험의 결과이다. 서빙 MECS 의 이웃 MECS 들 중 최소한의 통신 시간이 소요되는 가장 가까운 MECS 와 작업부하가 가장 적은 MECS 로 비교 실험했다. 부하가 적은 MECS 와 협업하면 전체적인 부하 균형이 맞으면서 시스템 처리량이 증가하여 연산 지연시간이 줄어든다. *P_Mig_L, P_Off_L* 는 서빙 MECS 에서 부하가 작은 MECS 로 파티셔닝하여 마이그레이션 또는 오프로딩하는 기법이다.

그리고 *P_Mig_C* 이 *P_Off_C* 보다 약 5%만큼 성능이 좋다. 차량에서 MECS 로 오프로딩하는 시간이 MECS 끼리 마이그레이션하는 시간보다 길기 때문이다. 이는 *P_Mig_L* 와 *P_Off_L* 에서도 마찬가지이다. 그리고 *P_Mig_C*, *P_Mig_L* 와 *P_Off_C*, *P_Off_L* 를 각기 비교하면 어떤 MECS 와 협업하는지가 성능에 영향 미친다는 것을 보여줬다. 거리적으로 가까운 것보다는 부하가 적은 MECS 와 협업하는 것이 약 9%만큼 성능이 좋다. 하지만, *P_Mig_C* 와 *P_Off_L* 에서 보듯이 어떤 MECS 와 협업하는지에 따라 성능이 달라질 수 있기 때문에 마이그레이션 파티셔닝 기법이 오프로딩 파티셔닝 기법보다 항상 좋은 성능은 아니다.



(그림 4) 파티셔닝 기법들의 차량 에너지 소비량 비교

그림 4 에서는 차량이 오프로딩할 때 소비하는 에너지량으로 파티셔닝 기법들의 성능을 비교하였다. *P_Mig_C* 와 *P_Mig_L* 는 태스크 전체를 오프로딩하기 때문에 차량이 소비하는 에너지량은 대체로 비슷하다. 그래서 마이그레이션을 이용한 파티셔닝 기법의 경우에 그림 3 처럼 어떤 MECS 와 협업할지 선택하는 것이 성능에 영향을 미친다. *P_Off_L* 는 *P_Off_C* 보다 약 6.6%만큼 더 많은 에너지를 소비한다. 그러나 그림 3 에서는 *P_Off_L* 이 더 좋은 성능을 보였다. 그래서 오프로딩을 이용한 파티셔닝 기법의 경우, 좋은 성능을 위해 서비스 지연시간과 차량의 에너지 소비량을 적절하게 조절해야 한다. 마이그레이션과 오프로딩 기법들을 비교하면, 태스크를 한 번에 오프로딩 하는 것보다는 파티셔닝하여 여러 번 오프로딩하는 것이 더 많은 에너지를 소비한다는 것을 확인할 수 있다.

5. 결론

태스크를 여러 개의 서브 태스크로 파티셔닝하여 다수의 MECS 에서 공동으로 연산하는 것이 단독적으로 수행할 때 보다 연산에 대한 부담을 덜어 전체적인 지연시간을 줄여준다. 본 논문에서는 오프로딩과 마이그레이션을 이용한 파티셔닝 기법들을 서비스 지연시간과 차량의 에너지 소비량 측면에서 성능 비교를 하였다. 대체적으로 마이그레이션 기법이 오프로딩 기법보다 지연시간에 대한 성능이 좋았지만, 어떤 MECS 와 협업하는지에 따라 성능이 달라진다. 차량의 에너지 소비량 측면에서는 여러 번 오프로딩하는 오프로딩 파티셔닝 기법의 성능이 안 좋았다. 향후 연구에서는 보다 다양한 환경에서 실험을 진행하고, 보다 다양한 측면에서의 성능 분석을 진행하고자 한다.

사사문구

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2021R1F1A1047113).

참고문헌

[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, Feb. 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, Apr. 2018.

[3] 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, Aug. 2021.

[4] Z. Cheng, M. Min, M. Liwang, L. Huang and Z. Gao, "Multi-Agent DDPG-Based Joint Task Partitioning and Power Control in Fog Computing Networks," *IEEE Internet of Things Journal (Early Access)*, Jun. 2021.

[5] 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, Dec. 2020.

[6] 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, Aug. 2018.