논문 21-46-02-15 The Journal of Korean Institute of Communications and Information Sciences '21-02 Vol.46 No.02
https://doi.org/10.7840/kics.2021.46.2.310

모바일 엣지 컴퓨팅 환경에서의 태스크 오프로딩 알고리즘

이 은 희•, 이 수 경°

Task Offloading Algorithm for Mobile Edge Computing

Eun-hee Lee•, Su-kyung Lee°

요 약

본 논문에서는 Mobile Edge Computing (MEC)에서 고연산 및 엄격한 지연 요구사항을 만족시켜야 하는 사용자 서비스를 지원하기 위해 Mobile Edge Network(MEN)로의 태스크 오프로딩 알고리즘을 제안한다. 제안된 알고리즘은 태스크 의존성 및 사용자 이동성을 고려한다. 시뮬레이션을 통해 제안된 알고리즘이 기존 연구 대비 개선됨을 보여준다.

Key Words : Task Offloading, Task Dependency, User Mobility, Mobile Edge Network

ABSTRACT

In this paper, we propose a task offloading algorithm to MEN to provide fast execution to computation-intensive and delay-sensitive applications in MEC. The proposed algorithm takes into account task dependencies and user mobility. We show that the algorithm proposed through simulation is improved compared to existing research.

Ⅰ. 서 론

최근 사용자들의 다양한 서비스 요구 증가에 따라 고성능이 요구되는 모바일 응용(애플리케이션)이 생겨나고 있다. 그러나 사용자 단말의 제한된 배터리 용량과 연산 처리 능력은 고성능이 요구되는 실시간 응용 실행에 적합하지 않다 [1, 6]. 이에 따라 사용자 단말과 가까운 MEN으로의 태스크 오프로딩 기법에 대한 연구가 많이 진행되고 있다[1-6].

특히 응용은 상호 의존성을 가지는 여러 개의 태스크로 나눌 수 있기 때문에 태스크 의존성을 고려하는 태스크 오프로딩 기법에 대한 연구가 많이 진행되었다 [3,4].

그러나 MEN으로의 태스크 오프로딩에서 모바일 단말을 지닌 사용자의 이동성은 small cell Base Station(sBS)의 부하를 변화시켜, 예측된 태스크 실행 시간에 영향을 미친다. 즉, 사용자의 이동성으로 인해 sBS에 연결되는 사용자 수와 함께, sBS의 부하도 변화하게 된다 [2]. 이에 따라 사용자 이동성을 고려한 태스크 오프로딩 기법에 대한 연구도 진행되었다 [2,5].

기존의 연구는 태스크 의존성 또는 사용자 이동성을 고려하였으나 앞서 언급한 바와 같이 상호 의존성을 가지는 태스크들로 구성된 모바일 응용이 증가하고 있으므로 태스크 의존성과 사용자 이동성을 모두 고려할 필요가 있다.

따라서 본 논문에서는 태스크 의존성과 사용자 이동성을 모두 고려하여 응용 실행 시간 단축을 위한 태스크 오프로딩 기법을 제안하고자 한다.

Ⅱ. 시스템 모델

본 논문에서는 연산과 저장이 가능한 여러 개의 sBS로 이루어진 MEN 환경을 가정한다. sBS들은 메쉬 토폴로지 형태로 연결되어 있으며 중앙 제어 장치와는 모두 연결되어 있다. 각 sBS는 한 번에 하나의 태스크만 실행이 가능하다고 가정한다[2].

본 연구에서는 실행해야 하는 응용을 가지는 여러 사용자가 MEN 내에서 이동하고 있는 환경을 가정한다. 그리고 각 사용자 이동성은 다중 커널 학습법 혹은 Lagrange's 보간법을 이용하여 예측된다[7,8]. 예측된 사용자 이동성은 사용자가 지나가는 경로의 sBS들로 표기된다. 예를 들어 사용자가 sBS1, 2, 4를 차례로 이동할 때, 사용자 이동성 Mu는 {S1, S2, S4}로 표기한다.

모든 사용자는 실행해야 할 응용을 하나씩 지니고 있다고 가정한다. 응용은 의존성을 가지는 여러 개의

※ 본 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1A2C1086191)
• First Author : (0000-0003-4846-7497)Yonsei University Department of Computer Science, oleh8823@gmail.com, 학생(석사), 회원
° Corresponding Author : (0000-0002-3497-3295)Yonsei University Department of Computer Science, sklee@yonsei.ac.kr, 정교수, 종신회원
논문번호 : 202001-251-B-LU, Received October 8, 2020; Revised November 11, 2020; Accepted November 13, 2020

태스크로 나눌 수 있으며, 이 의존성은 DAG(Directed Acyclic Graph)인 G=(V, E)로 표기할 수 있다[3].

MEN 내에는 실행해야 하는 응용들의 집합은 $A=\{A_1, A_2, \cdots, A_n\}$ 으로 표기할 수 있다.

Ⅲ. 태스크 오프로딩 기법

3.1 태스크 실행 시간 및 이동 시간

태스크는 t가 sBS s에서 실행될 때의 실행 시간은 식 (1)과 같이 계산될 수 있다.

$$E = \frac{c_t}{f_s} \tag{1}$$

c_t는 태스크 t를 실행하기 위해 필요한 CPU 사이클 수를, f_s는 태스크를 실행하는 sBS s의 CPU 주파수를 의미한다.

태스크 이동 시간은 태스크의 입력 및 출력 데이터 전송에 소요되는 시간을 의미한다. 태스크 A와 B 사이에는 의존성이 있고, 태스크 A와 B가 각각 sBS i, j에서 실행될 때 이동 시간은 태스크 A의 결과 데이터가 sBS i에서 sBS j로 이동되는 시간을 의미며, 식 (2)와 같이 계산된다.

$$T_{i,j}^{A,B} = \begin{cases} 0, & i = j \\ \dfrac{do_A}{rt_{i,j}}, & i \neq j \end{cases} \tag{2}$$

sBS i와 j가 같다면 태스크 이동 시간은 0으로 가정한다[3]. do_A는 태스크 A의 결과 데이터양이며, $rt_{i,j}$는 sBS i와 j 사이의 데이터 전송률을 의미한다.

3.2 태스크 우선순위

모든 태스크들은 서로 다른 우선순위를 가진다. 태스크 t의 우선순위는 식 (3)으로 계산되며, 태스크 오프로딩은 우선순위가 높은 태스크부터 결정한다.

$$rank(t) = \overline{E_{s \in Mu}} + \max_{t_A \in succ(t)} \{rank(t_A)\} \tag{3}$$

첫 번째 항은 태스크 t가 사용자 이동성 내의 sBS들에서의 실행 시간의 평균을 의미한다. $succ(t)$는 태스크 t 실행이 성공한 이후 실행해야 하는 태스크들의 집합으로 두 번째 항은 t 이후 실행하는 태스크들 중 우선순위 값이 가장 큰 값을 택하는 것을 의미한다.

3.3 태스크 실행 시작 및 종료 시각

태스크가 sBS에서 실행될 때 예상되는 실행 시작 시각(Estimated Start Time;EST) 및 종료 시각(Estimated Finish Time;EFT)은 실행되는 sBS에 따라 달라진다. 태스크 A가 sBS i에서 실행될 때 예상되는 EST는 식 (4)와 같이 계산된다.

$$EST(t,i) = \max_{t_A \in pred(t)} \left\{ AFT(t_A) + T_{s,i}^{t_A,t} \right\} \tag{4}$$

태스크 실행 시작 전 수행해야 하는 태스크들(pred) 중 실행 완료 시각(Actual Finish Time; AFT)과 태스크 이동 시간의 합 중 큰 값이 된다. EFT는 태스크 실행 시작 시각(식 (4))과 태스크 실행 시간의 합(식 (1))으로 계산된다.

태스크 오프로딩 기법에 의해 오프로딩이 결정되면 해당 EFT는 AFT가 된다. 본 논문에서는 MEN 내에서 실행해야 하는 모든 응용들(A)의 AFT를 최소화하고자 한다.

3.4 태스크 의존성 및 이동성 기반 오프로딩

태스크 오프로딩 기법은 하나의 태스크와 하나의 sBS가 있는 환경에서 로컬(단말) 혹은 sBS 실행 여부로 간단히 하여 0-1 knapsack 문제로 줄일 수 있어, NP-hard 문제이다. 따라서 다음과 같은 휴리스틱 알고리즘을 제안한다.

중앙 제어장치는 아래의 알고리즘을 실행하여 태스크 오프로딩을 결정하고 sBS에게 전달하여 태스크가 실행될 수 있도록 한다. 먼저 식 (3)을 이용하여 모든 태스크의 우선순위를 계산한 후, 우선순위를 기준으로 하여 태스크를 내림차순 정렬한다. 정렬된 태스크들은 차례로 다음을 반복한다. 사용자 이동성(Mu) 내에 있는 sBS에서 태스크를 실행했을 때의 EST(식 (4))와 태스크 실행 시간(식 (1))을 더하여 EFT를 계산한다.

Task Offloading Algorithm

```
1: Calculate the rank of all tasks, using (3)
2: Task List = {Sort rank values in decending order}
3: while(Task List is not null) do
4:    t = The first task of Task List
5:    EFT List = {EFT(t, p)}, p ∈ Mu (Equation
      (4)+(1))
6:   AFT(t) = min{EFT List}
7:   Delete t from the Task List
8: done
```

이들 중 가장 작은 EFT를 가지는 sBS로 태스크 오프로딩을 결정한다. 이는 각 태스크들이 이동성 내의 sBS로만 오프로딩 되도록 하여 사용자 이동성을 반영하며, [3]의 Centralized EFT Algorithm (CEFT)에서 MEN 내의 모든 sBS에서의 EFT를 고려하는 것과 대비된다. 모든 태스크들의 오프로딩이 결정될 때까지 위 과정을 반복한다.

이 알고리즘의 실행 시간은 태스크의 수가 m개 이고 sBS의 수가 n개 일 때, 최대 O(mn)이다.

Ⅳ. 시뮬레이션 결과

섹션 Ⅲ.에서 제안한 알고리즘을 검증하기 위해 시뮬레이션을 진행하였다. 시뮬레이션에 사용된 주요 파라미터들은 표 1과 같다. 각 응용은 10개의 태스크로 나눌 수 있다고 가정하고 DAG는 랜덤으로 생성하였다 [3]. 또한 사용자 이동성은 [7], [8]에 의해 예측된 경로를 따라 일정한 속도(1m/s)로 움직인다고 가정한다. 제안 알고리즘과 로컬(단말)에서의 실행(Local) 및 사용자 이동성 고려는 없으나 태스크 의존성을 고려한 [3]의 CEFT와 비교하였으며, 시뮬레이션 설정별로 1000번씩 반복하여 평균값을 그림으로 작성하였다. 그림 1은 사용자 수는 200명으로 고정하고 sBS 수의 증가 시, 제안 알고리즘은 Local 및 CEFT보다 평균적으로 각각 30.4%, 11.8%의 AFT 개선율을 보였다. sBS의 수가 증가할수록 성능이 향상함을 알 수 있다. 그림 2는 sBS 수는 10개로 고정하고 사용자 수 증가 시, 제안 알고리즘은 Local 및 CEFT보다 평균적으로 각각 38.8%, 16.3%의 AFT가 개선율을 보였으며, 사용자 수가 적을수록 성능이 좋아짐을 알 수 있다.

표 1. 시뮬레이션 설정값 [2]
Table 1. Simulation Settings

Parameter	Value/Range
sBS uplink/downlink	80Mbps/80Mbps
Intra MEN transmission rate	1000Mbps
f_s & Instruction Set	2GHz & CISC
f_l & Instruction Set	2GHz & RISC
c_t	2×10^8 ~ 2×10^9
Input data size	1~30MB

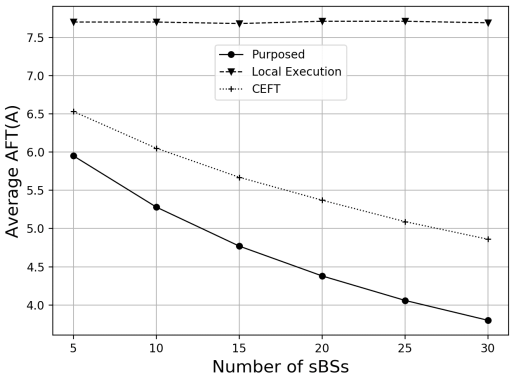

그림 1. sBS 수에 따른 AFT 평균값
Fig. 1. average AFT(A) vs. number of sBSs

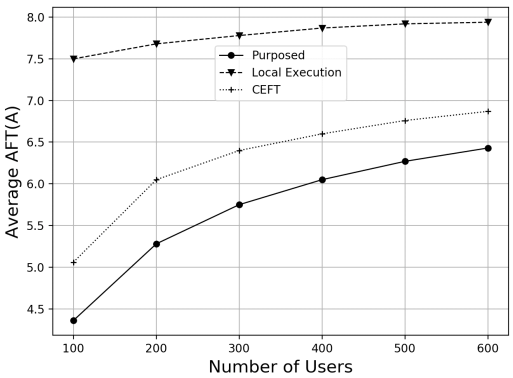

그림 2. 사용자 수에 따른 AFT 평균값
Fig. 2. average AFT(A) vs. number of users

Ⅴ. 결 론

본 논문에서는 MEN 사용자들에게 응용의 빠른 실행을 제공하기 위한 태스크 오프로딩 기법을 제안하였다. 제안 기법에서는 태스크 의존성과 사용자 이동성을 모두 고려하였다. 시뮬레이션을 통해 제안 알고리즘이 사용자 이동성 고려가 없는 기존 연구보다 AFT 측면의 성능 개선을 증명하였다.

References

[1] Y. Tae, J. Lee, and S. Pack, "A study on the task offloading scheme considering execution delay in edge computing," in *Proc. Symp. KICS*, pp. 148-149, Pyeongchang, Korea, Jan. 2019.

[2] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni

and R. Wang, “User mobility aware task assignment for mobile edge computing,” *FGCS*, vol. 85, pp. 1-8, Aug. 2018.

[3] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, “Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach,” *IEEE IoT J.*, vol. 7, no. 3, pp. 1678-1689, Mar. 2020.

[4] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, “Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency,” *IEEE TWC*, vol. 19, no. 1, pp. 235-250, Jan. 2020.

[5] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu and H. Duan, “Mobility-aware multi-user offloading optimization for mobile edge computing,” *IEEE TVT*, vol. 69, no. 3, pp. 3341-3356, Mar. 2020.

[6] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” *IEEE COMST*, vol. 19, no. 3, pp. 1628-1656, Mar. 2017.

[7] T. M. T. Do, O. Dousse, M. Miettinen, and D. Gatica-Perez, “A probabilistic kernel method for human mobility prediction with smart-phones,” *PMC,* vol. 20, pp. 13-28, Jul. 2015.

[8] B. Li, H. Zhang, and H. Lu, “User mobility prediction based on Lagrange’s interpolation in ultra-dense networks, in: personal, indooor, and mobile radio communications,” *IEEE PIMRC 2016*, pp. 1-6, Valencia, Spain, Sep. 2016.