https://doi.org/10.7840/kics.2024.49.4.547

저궤도 위성 네트워크에서의 엣지 컴퓨팅을 위한 동적 코드 오프로딩 및 CPU 클럭 스케일링

김 정 환◆, 곽 정 호°

Dynamic Offloading and CPU Clock Scaling for Edge Computing in LEO Satellite Networks

Jeong-hwan Kim◆, Jeong-ho Kwak°

요 약

저궤도 (LEO) 위성 통신은 기존 지상 네트워크의 글로벌 커버리지 구축 문제를 해결하는 좋은 방법으로 채택되고 있다. 또한 6G 서비스 시대로 나아감에 있어 복잡한 어플리케이션의 출현은 자연스럽게 장치들의 계산 부담을 증가하게 만들었고, 합리적인 컴퓨팅을 위해 오프로딩 및 엣지 컴퓨팅 기법이 활발하게 적용되고 있다. 따라서 6G 서비스의 전 지구적이고 원활한 활용을 위해, 저궤도 위성에 컴퓨팅이 가능한 서버를 설치하여 엣지 컴퓨팅을 가능하게 하는 위성 엣지 컴퓨팅 (SEC) 연구가 활발히 진행되고 있다. 다만 진행 중인 연구들은 지상 네트워크에서의 엣지 컴퓨팅 기법을 위성 네트워크로 확장만 할 뿐, 위성 네트워크의 물리적인 토폴로지 변화를 고려한 동적인 채널 상태를 합리적으로 고려하지 않고 있다. 이를 해결하기 위해 본 연구에서는 동적 채널 상태를 합리적으로 반영하는 저궤도 위성 네트워크에서의 엣지 컴퓨팅 프레임워크를 제안하고, 시스템의 전력 소모와 전파 지연 최소화를 목표로 하는 Lyapunov 최적화 기법 기반의 알고리즘을 제안한다. 제안한 SEC 시나리오에서의 시뮬레이션을 통해 알고리즘이 동적으로 변화하는 채널 상태와 요청되는 서비스에 따라 적응적으로 동작하여 기존의 통상적인 방법들에 비해 같은 처리 지연 대비 더 적은 전력을 소비함을 확인할 수 있다.

키워드 : Lyapunov 최적화, 위성 엣지 컴퓨팅, 저궤도 위성 통신
Key Words : Lyapunov optimization, Satellite edge computing, LEO satellite communication

ABSTRACT

Low Earth Orbit (LEO) satellite communication is an effective solution for addressing global coverage challenges in terrestrial networks. As we enter the 6G era, the rise of complex applications has increased device computational demands, driving the adoption of techniques like offloading and Edge Computing for efficient processing. To achieve seamless global 6G service integration, active research is underway in Satellite Edge Computing (SEC). SEC involves deploying computation-capable servers on LEO satellites, enabling edge computing. However, many studies extend terrestrial Edge Computing techniques to satellite networks without adequately considering dynamic channel conditions due to physical topology changes. Our research proposes an

※ 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업(IITP-2023-2018-0-01424)의 연구결과로 수행되었습니다.
◆ First Author: Daegu Gyeongbuk Institute of Science and Technology Department of Electrical Engineering and Computer Science, ghks9876@dgist.ac.kr, 학생회원
° Corresponding Author: Daegu Gyeongbuk Institute of Science and Technology Department of Electrical Engineering and Computer Science, jeongho.kwak@dgist.ac.kr, 종신회원
논문번호 : 2023-11-144, Received November 17, 2023; Revised December 28, 2023; Accepted December 29, 2023

Edge Computing framework tailored to LEO satellite networks, effectively addressing dynamic channel conditions. We introduce optimization algorithms to minimize system power consumption and propagation latency. Simulations in our proposed SEC scenario confirm our algorithm's adaptability to changing channel conditions and service requests, resulting in lower power consumption compared to conventional methods for the same processing latency.

Ⅰ. 서 론

급격한 인터넷 산업의 성장에도 불구하고 저개발 지역까지 포함하는 글로벌 커버리지 달성문제는 아직도 해결해야 할 도전과제로 남아있다. 게다가 자연 재해로부터 영향을 받기 쉬운 지상 네트워크 시스템은 비상 상황에서 심각한 통신 장애를 겪을 수 있다. 이 맥락에서 위성 통신은 광범위한 커버리지, 풍부한 스펙트럼 및 최소한의 간섭[1]을 통해 전통적인 지상 네트워크의 취약성을 보완하기 위한 잠재적인 해결책 중 하나로 큰 주목을 받고 있다[2]. 그러나 6G 시대로의 전환은 저지연과 고속 통신 서비스의 제공을 필요로 하는데 기존의 정지 궤도 위성은 상당한 경로 손실과 지연 문제로 인해 원활한 서비스 제공에 어려움을 겪고 있다. 스팟 빔 안테나 및 레이저 전송과 같은 기술의 발전은 위성 통신, 특히 저궤도 위성 통신을 효율적이고 간결하게 변화시켰으며 지상국까지의 지연 시간을 크게 줄이게 되었다.

스마트폰과 태블릿에 의한 연산 집약적인 애플리케이션의 급증은 음성 인식부터 멀티미디어 처리까지 다양한 Quality of Service (QoS) 요구를 촉발시켰다. 클라우드 컴퓨팅은 이러한 요구 사항을 잠재적으로 해결할 수 있지만 지상 네트워크가 제약을 받는 지역에서는 그 효과가 극히 제한되게 된다. 이러한 물리적 제약을 완화하기 위해 도입된 모바일 엣지 컴퓨팅(MEC)은 연산 및 저장 서비스를 근처 지역 서버에게 요청하여 지연과 소모 전력을 절약하고 QoS를 크게 향상시킬 수 있다[3]. 자연스럽게 MEC와 위성 네트워크의 융합은 상당한 관심을 받았으며[4]-[6], 저궤도 위성 산업의 급격한 확장과 상기한 기술의 진보로 인해 이 연구 분야는 점점 더 실현 가능성이 높아지고 있다. 그러나 현재의 연구 대부분은 기존 MEC 기술의 위성 네트워크로의 확장에만 중점을 두고 있으며 통합된 합리적인 아키텍처 설계가 부족하다. 특히 최근에 소개된 저궤도 위성 네트워크를 활용한 엣지 컴퓨팅 연구[7]에서도 지상망에서의 최적화와 유사하게 고정된 상황에서 정적인 최적화를 시도하는 것을 볼 수 있다. 이는 채널 상태가 거의 변하지 않는 지상망의 특징이 그대로 보존된 것인데 저궤도 위성 네트워크에서도 유사하게 적용하는 것은 현실적이지 못하다. 낮은 고도의 저궤도 위성은 그만큼 빠른 속도로 지구를 공전하게 되고, 이에 따라 지상과 저궤도 위성 사이 채널 또한 매우 동적으로 변하기 때문이다. 결론적으로 저궤도 위성 네트워크에 관해 현실적으로 연구를 진행하려면 저궤도 위성의 고유 특성인 빠른 이동성과 이에 따른 동적인 채널 상태를 반영해야 한다.

이러한 문제를 해결하기 위해, 우리는 지상-우주 엣지 컴퓨팅 시스템에 대한 포괄적인 새로운 프레임워크를 제안하고, 전력 소비와 전파 지연의 평균 가중합을 최소화하기 위한 문제를 정의한다. 정의한 문제를 풀기 위해 Lyapunov 최적화 이론을 사용하여 매 타임 슬롯마다의 동적인 오프로딩 및 CPU 스케일링 알고리즘을 개발하고, 이를 제안된 저궤도 위성 네트워크 엣지 컴퓨팅 프레임워크 상에서 시뮬레이션을 통해 평가한다. 이러한 결과는 기존의 통상적인 알고리즘들과 비교했을 때 전파 지연, 대기열 길이 및 전력 효율성 측면에서 상당한 개선을 보여주고 있음을 확인할 수 있다.

Ⅱ. 시스템 모델

시나리오 모델: Fig. 1은 사용자들의 계산 요구를 충족시키기 위해 설계된 지상-우주 통합 엣지 컴퓨팅 프레임워크를 보여준다. 지상에는 $\mathcal{I} = \{1, \cdots, I\}$로 표시된 I명의 모바일 사용자들이 있고, 이러한 사용자들은 타겟 지역 전역에 무작위로 분포되어 계산 집약적인 요청을 지속적으로 생성한다. 우주에는 여러 개의 LEO 위성이 타겟 지역 위를 지나가고 동일한 궤도를 공유한다. 이러한 LEO 위성들의 통신을 조정하고 중앙 제어를 하기 위해 지상 사용자들 근처에 하나의 게이트웨이 (GW)가 있다고 가정한다. 연결된 하나의 위성이 타겟 지역에서 벗어날 때, 동일한 궤도의 다음 위성에게 작업을 넘기게 되고, 이러한 LEO 위성은 GW와 협력하여 사용자들에게 비디오 분석과 같은 계산 작업을 지원하게 된다. 네트워크 기능 가상화 (NFV) 및 소프트웨어 정의 네트워크 (SDN) 기술을

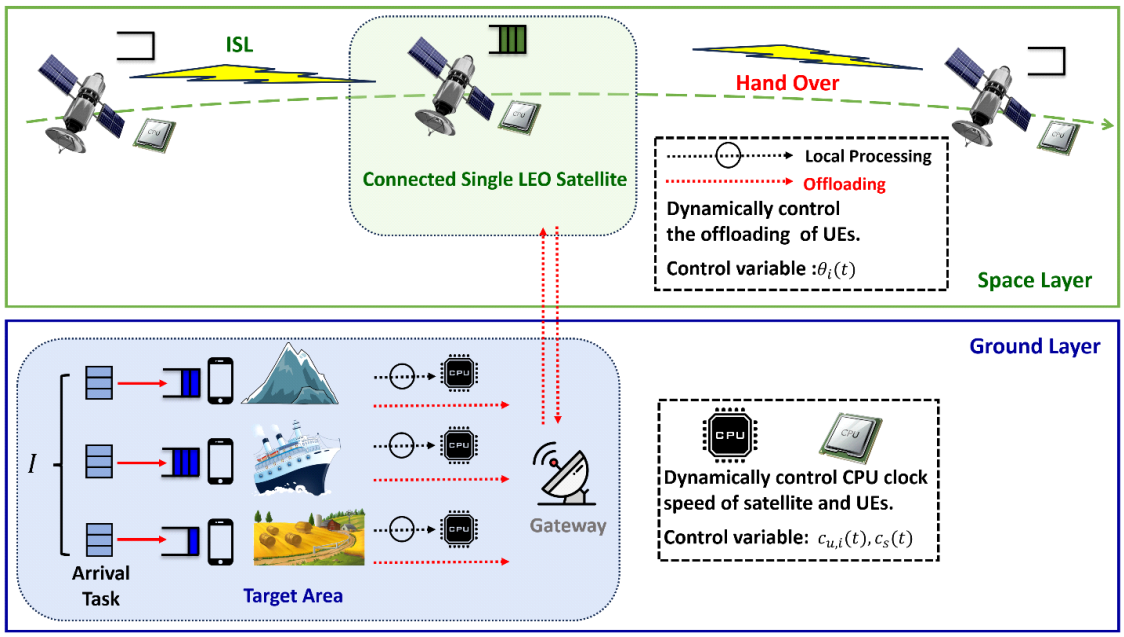

Fig. 1. Dynamic Edge Computing Framework on LEO Satellite Networks

통해 프레임워크 상 모든 장치들에 대한 중앙 제어를 할 수 있게 되고, 이러한 SDN 컨트롤러는 GW에 배치될 수 있으며 GW가 LEO 위성의 채널 상태 정보 (CSI) 및 계산 작업 정보와 같은 필수 시스템 정보에 액세스할 수 있다고 가정한다. 이에 기반하여 제안하는 프레임워크에서도 알고리즘 실행은 GW에서 이루어진다. 결과적으로, 사용자들은 다음과 같은 두 단계를 통해 그들의 요청된 계산 태스크를 위성에게 오프로딩 하게 된다. 먼저, C 밴드 대역 무선 백홀 링크를 통해 태스크를 GW로 보낸다. 그리고 GW는 이러한 태스크를 Ka 밴드 대역을 통해 LEO 위성으로 오프로딩 한다. 타임 슬롯 t에 GW가 LEO 위성으로 전송 가능한 최대 채널 용량 $r(t)$는 shannon capacity 공식을 사용하여 다음과 같이 표현할 수 있다.

$$r(t) = B_{Ka}\log_2(1+\frac{P_{gw}^N|h(t)|^2}{N_0}). \quad (1)$$

B_{Ka}는 Ka 밴드 대역에서의 대역폭, P_{gw}^N 는 GW의 전송 전력, N_0는 잡음 전력, $h(t)$는 타임 슬롯 t일 때의 GW와 LEO 위성 간 채널 상태를 나타낸다. 채널 상태엔 거리에 따른 경로 손실, 페이딩, 강우 감쇠 등이 고려된다. 이 최대 채널 용량을 사용자들의 오프로딩 여부에 따라 오프로딩하는 각 사용자들이 $r'(t)$만큼 나눠 사용할 수 있다. LEO 위성에서 컴퓨팅이 완료되면 그 결과 데이터는 사용자들에게 역순으로 다시 전송된다. 이 과정에서 타임 슬롯 t에서의 사용자 i 오프로딩 여부를 나타내는 매개 변수로 $\theta_i(t) = (0, 1)$을 정의한다. 1일 땐 오프로딩, 0일 땐 오프로딩 하지 않고 각 사용자의 장치에서 처리함을 나타낸다. 각 사용자들의 장치와 LEO 위성은 단일 안테나, GW는 멀티 안테나를 갖춘 환경에서 각 사용자들은 GW에 통신하기 위해 직교 주파수 분할 다중 접속 (OFDMA) 방식을 사용한다.

컴퓨팅 및 네트워킹 모델: P_u^N 및 P_{gw}^N는 각각 사용자 장치와 GW의 고정된 송신 전력을 나타낸다. 실제로 지상 위 장치의 송신 전력은 채널 상태에 따라 약간의 변동을 나타낼 수 있으며, 채널 상태의 함수로 전력 소비를 정확하게 모델링하는 것은 매우 어렵다. 따라서, 송신 전력 모델링과 관련된 많은 연구들은 대부분 고정된 송신 전력의 가정을 채택하고 있다[8].

컴퓨팅 모델에 대해 사용자의 모바일 장치는 단일 코어 CPU, LEO 위성의 경우 멀티 코어 CPU를 갖추고 있다고 가정한다. 이러한 CPU들은 여러 태스크들을 순차적으로 처리하는 능력을 가지고 있다. 사용자 장치의 CPU를 포함한 대부분의 현대 프로세서는 동적 전압 및 주파수 조절 (DVFS) 능력을 갖추고 있어서 각 타임 슬롯 t동안 다음과 같이 CPU 클럭 속도를 동적으로 조절할 수 있다. $c(t) \in \{c_1(t), \cdots, c_{max}(t)\}$ (cycles/ Δt). 통상적인 DVFS CPU 전력 소비 모델은

다음과 같다. $p^p(c(t)) = \alpha_1 c(t)^3 + \alpha_2 c(t)^2 + \alpha_3 c(t) + \alpha_4$, $\forall t \in \mathcal{T}$

대기열 모델: 지상의 타겟 지역에서, 타임 슬롯 t에 a(t) = ($a_1(t)$, ⋯, $a_I(t)$) (bits)의 태스크가 생성되어 각 사용자 장치 대기열에 들어오게 된다. $a(t)$는 모든 타임 슬롯에서 독립적으로 동일하게 분포되며 (i.i.d. 환경), 그 기대값은 $\mathbb{E}[a_i(t)] = \lambda_u \geq 0$으로 표시된다. 모든 타임 슬롯에서 도착되는 모든 태스크는 다음과 같이 제한된다. $a_i(t) \leq a_{max}$. 또한, 아직 처리되지 않거나 오프로딩 되지 않는 태스크는 각 사용자 장치 대기열에 저장된다. Fig. 1에서 본 것처럼 다음과 같이 정의된 비트로 측정된 두 가지 유형의 작업량 대기열이 있다:

$$Q_{u,i}(t+1) = \left[Q_{u,i}(t) - \frac{\left(1-\theta_i(t)\right)c_{u,i}(t)}{\gamma} - \theta_i(t) r'(t) + a_i(t) \right]^+, \quad (2)$$

$$Q_s(t+1) = \left[Q_s(t) - N_c \frac{c_s(t)}{\gamma} + \sum_{i=1}^{I} \theta_i(t) r'(t) \right]^+. \quad (3)$$

γ는 태스크의 단일 비트를 처리하는데 필요한 CPU 클럭 사이클 수이고, $r'(t)$는 타임 슬롯 t일 때 한 명의 사용자가 사용 가능한 채널 용량, N_c는 LEO 위성 CPU의 멀티 코어 개수이다.

$Q_{u,i}(t)$, $Q_s(t)$는 타임 슬롯 t일 때 사용자 i와 LEO 위성의 처리 대기열이고, $[x]^+ = max(x, 0)$을 나타낸다. 타임 슬롯 $t+1$ 일 때의 사용자 대기열은 타임 슬롯 t일 때의 사용자 대기열에 생성되는 태스크 $a_i(t)$만큼 유입되고, 오프로딩 여부에 따라 $\frac{c_{u,i}(t)}{\gamma}$ (로컬에서 CPU로 처리) 또는 $r'(t)$ (오프로딩)만큼의 태스크가 처리된다. 이와 유사하게 타임 슬롯 $t+1$ 일 때의 LEO 위성 대기열은 타임 슬롯 t 일 때의 LEO 위성 대기열에 모든 사용자들의 오프로딩하는 태스크 합 ($\sum_{i=1}^{I} \theta_i(t) r'(t)$)만큼 유입되고, CPU 클럭 속도에 따라 항상 $N_c \frac{c_s(t)}{\gamma}$만큼 처리된다.

Ⅲ. 동적 오프로딩 및 CPU 스케일링 알고리즘

본 파트에서는 전력 소모 최소화, 전파 지연 및 대기열 안정성 유지를 포괄하는 최적화 문제 정의를 제시한다. 오프로딩 결정을 할 때, 지상 네트워크와는 달리 LEO 위성까지의 거리가 상당히 긴 점을 고려해야 한다. 따라서, 태스크 전송 중 발생하는 전파 지연이 중요한 요소로 작용하게 되고, 이러한 전파 지연을 오프로딩 결정의 페널티로 최적화 문제에 포함시킨다. 제안하는 타임 슬롯 t에서의 시스템 전파 지연은 다음과 같이 표현된다.

$$L_{sys}(t) = 2 \sum_{i=1}^{I} \theta_i(t) \frac{d(t)}{l}. \quad (4)$$

$d(t)$는 타임 슬롯 t일 때 GW와 LEO 위성 간 거리이고, l은 빛의 속력이다.

문제 정의: 제안하는 알고리즘의 목표는 오프로딩 동작 여부와 사용자들 및 LEO 위성의 CPU 클록 속도를 조절해서 시스템 전력 소모 및 전파 지연을 최소화하는 것이다. 사용자들에게 유입되는 태스크들은 시스템이 유한한 기간 내에 처리할 수 있는 범위 내에서 생성된다. 상기의 목표를 가진 장기적인 관점에서의 최적화 문제를 다음과 같이 정의한다.

$$(\mathbf{P}): \min_{\boldsymbol{\theta}, \boldsymbol{c_u}, \boldsymbol{c_s}} \left(\lim_{T\to\infty} \frac{1}{T} \sum_{i=1}^{I} \mathbb{E}\{ \boldsymbol{P_{sys}(t)} + \boldsymbol{w L_{sys}(t)} \} \right), \quad (5)$$

$$s.t.\ (C1): \limsup_{T\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left\{ \sum_{i=1}^{I} Q_{u,i}(\tau) + Q_s(\tau) \right\} < \infty. \quad (6)$$

w는 시스템에서의 전력 소모와 전파 지연의 비율을 조절하는 스케일링 변수이고, $P_{sys}(t)$는 타임 슬롯 t일 때 시스템에서 소모하는 모든 전력의 합이며, $(\boldsymbol{\theta}, \boldsymbol{c_u}, \boldsymbol{c_s}) \triangleq \left(\boldsymbol{\theta(t)}, \boldsymbol{c_u(t)}, c_s(t)\right)_{t=0}^{\infty}$을 의미한다. 제약 조건 (C1) 은 대기열 길이가 유한함 (서비스 시간이 유한함)을 의미한다.

Lyapunov 최적화 알고리즘 도출: Lyapunov drift-plus-penalty 기법을[9] 사용하여 알고리즘을 도출한다. 이 접근 방식은 생성되는 태스크의 정보와 미래의 네트워크 상태에 대한 정보가 필요하지 않고 현재의 대기열 길이와 처리량 정보만알아도 된다는 장점을 제공한다. 따라서 확률적 최적화 문제를 각 타임 슬롯마다의 결정론적 최적화 문제로 변환할 수 있다. 새로운 목적 함수를 얻기 위해 Lyapunov 함수와 Lyapunov drift 함수를 먼저 다음과 같이 정의한다.

$$L(t) \triangleq \frac{1}{2}\left\{\sum_{i=1}^{I} Q_{u,i}(t)^2 + Q_s(t)^2\right\}, \quad (7)$$

$$\Delta\big(L(t)\big) \triangleq \mathbb{E}\{L(t+1) - L(t) \mid \mathbf{Q}(t)\}. \quad (8)$$

여기서 $\mathbf{Q}(t) = \{Q_{u,1}(t), \cdots, Q_{u,I}(t), Q_s(t)\}$이다. Lyapunov 함수 (7)은 사용자들과 LEO 위성의 대기열 길이가 발산하지 않도록 하는 안정성을 보장하도록 설계되었다. 장기적 관점에서 목적함수 (5)를 최소화하기 위해, 매 타임 슬롯의 소모 전력과 전파 지연을 페널티로 반영한 Lyapunov drift-plus-penalty 함수를 다음과 같이 정의한다.

$$\Delta\big(L(t)\big) + V\mathbb{E}\{P_{sys}(t) + wL_{sys}(t) \mid \mathbf{Q}(t)\}. \quad (9)$$

V는 목적함수 (5)와 대기열 안정성 사이 균형을 조절할 수 있는 매개변수이다. 앞선 전개 과정을 통해, 매 타임 슬롯마다 기존의 목적함수 (5)를 최소화하는 것 대신 새로운 목적함수 (9)을 최소화하는 것으로 새로운 목표를 정의할 수 있다. 이제 정의한 작업량 대기열 모델 (2)-(3)와 생성되는 태스크, 새로운 목적함수 (9)을 이용하여 upper bound를 설계할 수 있다.

Lemma 1: 제어 변수 $(\theta_i(t) \in (0,1),\ c_{u,i}(t) \in \{c_{u,1}(t), \cdots, c_{s,\ max}(t)\}$을 통해 upper bound를 다음과 같이 전개한다.

$$\begin{aligned}
&\Delta\big(L(t)\big) + V\mathbb{E}\{P_{sys}(t) + wL_{sys}(t) \mid \mathbf{Q}(t)\} \\
&\le J + V\mathbb{E}\{P_{sys}(t) + wL_{sys}(t) \mid \mathbf{Q}(t)\} \\
&\quad - \sum_{i=1}^{I} \mathbb{E}\left\{\left(\frac{(1-\theta_i(t))c_{u,i}(t)}{\gamma} + \theta_i(t)r'(t)\right)Q_{u,i}(t) \mid \mathbf{Q}(t)\right\} \\
&\quad - \mathbb{E}\left\{\frac{N_c c_s(t)}{\gamma} Q_s(t) \mid \mathbf{Q}(t)\right\}.
\end{aligned} \quad (10)$$

이 때 $J = \frac{1}{2}\left(\sum_{i=1}^{I}\left\{\frac{c_{u,max}^2}{\gamma} + a_{max}^2 + 2\lambda_u Q_{u,i}(t)\right\} + \frac{N_c^2 c_{s,max}^2}{\gamma^2} + 2r^2(t) + 2r(t)Q_s(t)\right)$이다.

Proof: *Lemma*1 은 연구[10]에서 유도한 결과에 약간의 수학적인 기법을 가미하면 유도할 수 있다. ■

동적 오프로딩 및 CPU 스케일링 알고리즘: 새로이 정의한 목적함수 (10)의 상한을 최소화하는 알고리즘을 다음과 같이 전개할 수 있다. 알고리즘 1의 묘사를 통해 메커니즘을 소개한다. 본 논문의 전체적인 최적화 과정에서 매 타임 슬롯마다 우리가 제어하려는 변수는 사용자들의 오프로딩 여부, 사용자들과 LEO 위성의 CPU 클럭 속도 ($\theta(t)$, $c_u(t)$, $c_s(t)$)이다. 가능한 제어 변수 조합에서 유도한 upper bound (10)를 최소로 만드는 최적의 조합을 찾는 게 알고리즘 1의 목표이다. 모든 제어 변수 조합을 넣어 최적 조합을 찾는 것보다 알고리즘의 실행 시간을 감소시키기 위해, 알고리즘 1은 세 가지 단계로 진행하여 복잡도를 낮춘다. Upper bound (10)는 사용자마다 $\theta_i(t)$, 1-$\theta_i(t)$로 각각 묶어서 오프로딩 할 때 ($\theta_i(t) = 1$)의 항 $U_{i,\ off}(t)$, 오프로딩 하지 않을 때 ($\theta_i(t) = 0$)의 항 $U_{i,\ noff}(t)$으로 나눠 구분할 수 있다. 이 항들의 최소값, 최대값을 비교하여 어떤 조합에서도 무조건 오프로딩 또는 로컬에서의 처리를 선택하게 되는 사용자들을 찾는 게 첫 번째 단계이다. 이후 두 번째 단계에서는 첫 번째 단계에서 결정된 사용자들의 오프로딩 여부를 고정하고 나머지 가능한 오프로딩 여부 조합 ($\theta_{fixed}(t)$)에 대해 upper bound (9)를 최소로 만드는 사용자들과 LEO 위성의 CPU 클럭 속도 ($c_{u,i}(t)$, $c_s(t)$)를 결정한다. 그리고 각각을 비교하여 타임 슬롯 t에서의 최적 제어

Algorithm 1 동적 오프로딩 및 CPU 스케일링

At each time slot t, Given input parameters.
STEP 1: Find always offloading or non-offloading UEs.
for $i = 1 : I$ **do**
  if $\max U_{i,off}(t) < \min U_{i,noff}(t)$ **then**
    $\theta(i,t) = 1$.
  else if $\min U_{i,off}(t) < \max U_{i,noff}(t)$ **then**
    $\theta(i,t) = 0$.
  end if
end for
$\boldsymbol{\theta}_{\mathbf{fixed}}(:,t) = \boldsymbol{\theta}(:,t)$
STEP 2: Optimize for remaining UEs.
Find control variables that minimize upper bound for every possible $\boldsymbol{\theta}_{\mathbf{fixed}}(:,t)$.
STEP3: Update Queues.

Parameter	Value	Parameter	Value
γ	1000 cycles/bit	I	10
w	150	B_{Ka}	100 MHz
N_c	8	$c_{u,max}$	3.4 GHz
P_u^N	2 W	$c_{s,max}$	4.5 GHz
P_{gw}^N	20 W	λ_u	2.5 Mbits

Fig. 2. Simulation parameter

변수 조합 ($\theta(t)$, $c_u(t)$, $c_s(t)$)을 찾게 되는 것이다. 이 결과를 기반으로 다음 타임 슬롯 $t+1$의 각각의 작업량 대기열 모델 (2)-(3)을 업데이트 하는 게 마지막 세 번째 단계이다.

Ⅳ. 시뮬레이션

시뮬레이션 환경: 타겟 지역 위에서 하나의 LEO 위성이 GW와 연결되어 있으며, 이 위성은 550 km의 고도와 7.66 km/s의 속력으로 궤도를 따라 운행하고 있다. LEO 위성 토폴로지의 특성 상 위성의 위치는 고정되어 있지 않으며 매 타임 슬롯마다 변경되고, 위성이 타겟 지역을 벗어나게 되면 처리하던 태스크들을 같은 궤도의 다음 위성에게 핸드오버 하게 된다. 이 때 시나리오 상 LEO 위성-LEO 위성 간 통신은 광통신을 활용하여, 처리하던 태스크들을 다음 LEO 위성에게 핸드오버 하는 시간은 논문에서 사용한 한 타임 슬롯 (1sec)에 비해 매우 작은 값이라고 가정한다. LEO 위성 - GW 간 통신에는 29 GHz Ka 밴드 대역을 활용하고 잡음 전력 밀도는 통상적인 -174 dBm/Hz, 전체 타임 슬롯은 20,000 인 환경에서 시뮬레이션을 진행한다. 시뮬레이션에 사용된 파라미터 세팅은 다음과 같다.

시뮬레이션 결과: Fig. 3은 평균 대기열 길이와 평균 목적함수 간 트레이드오프 관계를 비교 알고리즘들과 함께 나타낸다. 균형을 조절하는 매개변수인 V 값의 변화에 따라 평균 대기열 길이와 목적함수 간의 비율은 다음과 같은 양상으로 변화한다. V가 증가하면 시스템이 목적함수 변화에 민감해져 평균 목적함수 값이 감소하게 되고, 반대로 대기열 길이 변화에는 덜 민감해져 평균 대기열 길이는 증가하게 된다 (점점 그래프의 오른쪽으로 이동하게 된다). V값은 시뮬레이션 환경에서 우리가 조절할 수 있는, 평균 목적함수와 평균 대기열 길이 사이의 비중을 조정하는 값이며, 실제로 물리적인 의미는 없는 (단위가 없는)가상의 값이다. 그래프에 직접적으로 나타나진 않지만 V값 조절에 따라 평균 목적함수 값이 크고 평균 대기열 길이가 작아질 수도 있고 (그래프의 왼쪽), 평균 목적함수 값이 작고 평균 대기열 길이가 커지게 (그래프의 오른쪽)만들 수도 있다. 너무 크거나 너무 작은 V값을 선택하면 극단적으로 조절하게 되어 평균 목적함수나 평균 대기열 길이가 발산할 수 있다. 시뮬레이션 환경을 설계할 때 평균 목적함수와 평균 대기열 길이 모두가 수렴하는 범위의 V값을 찾아 그 범위 내에서 진행하는 것이 중요하며, 본 논문의 시뮬레이션 환경도 이러한 V값 범위 내에서 진행되었다 ($5 \cdot 10^{15} \leq V \leq 8 \cdot 10^{16}$). Fig. 3에서 평균 목적함수-평균 대기열 길이 간 tradeoff 관계의 변화는 V가 충분히 크지 않은 범위 (e.g., $V \leq 8 \cdot 10^{15}$), 즉 평균 대기열 길이가 2 Gbits/Δt 보다 작은 구간에서 두드러지게 나타나며, 그 이상의 범위에서는 소극적으로 나타난다. 제안한 알고리즘은 Fig. 3에서 평균 대기열 길이가 2 Gbits/Δt 부근일 때 (V값이 $8 \cdot 10^{15}$ 부근일 때) 가장 좋은 성능을 보임을 확인할 수 있다. 비교 알고리즘들의 정보는 다음과 같다. 1) Selfish Users: 목적함수 (10)에 대해서 각 사용자들이 전체 시스템의 상한을 줄이기보다 자신들만의 목적함수를 줄이기 위해 제어 변수들을 조절하는 상황. 2) Offloading – DVFS: 모든 사용자들이 항상 오프로딩을 하지만 DVFS 기법을 활용하여 CPU 클럭 속도는 동적으로 조절하는 상황. 3) Offloading – Mid Frequency: 모든 사용자들이 항상 오프로딩하며 DVFS 기법을 활용하지 않고 고정된 CPU 클럭 속도를 사용한다. 제안한 알고리즘이 다른 비교 알고리즘들에 비해 모든 평균 대기열 길이 범위에서 최대 70%까지 개선된 성능을 보여준다.

Fig. 4는 다양한 w 값 상황에서의 V변화에 따른 평균 대기열 길이 – 평균 오프로딩 비율 관계를 나타낸

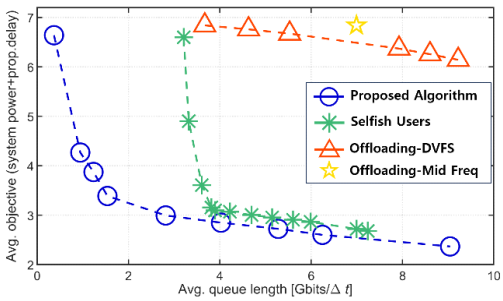

Fig. 3. Avg. queue length – Avg. objective tradeoff

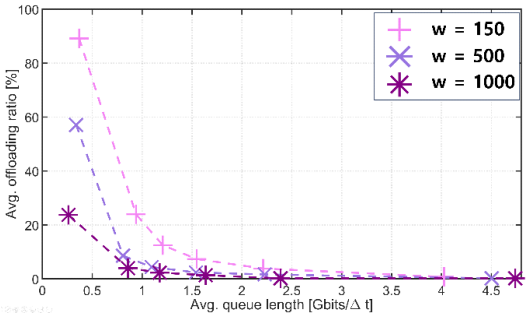

Fig. 4. Avg. queue length – Avg. offloading ratio

다. w는 기존의 목적함수 (5)에서의 시스템 전력 소모 - 전파 지연 사이 비율을 조절하는 매개변수이다. w가 증가하면 오프로딩의 페널티로 작용하는 전파 지연 변화에 더 민감하게 반응하게 되므로, 평균적인 오프로딩 선택 비율이 감소하게 된다. 이는 Fig. 4에서 w가 커질수록 점차 평균 오프로딩 비율이 감소하는 모습을 통해 확인할 수 있다. 또한 공통적으로 V가 증가할 때, 즉 시스템의 목적함수 변화에 민감하게 반응하게 될수록 평균 오프로딩 비율이 감소하는 것을 볼 수 있다.

Fig. 5와 Fig. 6은 w에 따른 사용자 장치와 LEO 위성의 평균 대기열 길이 - 평균 CPU 클럭 속도 관계를 나타낸다. 이는 앞선 Fig. 4와 연관 지어서 분석할 수 있다. 언급한대로 w의 증가는 더 적은 오프로딩 선택으로 이어지며 이는 사용자들의 장치에서 처리해야 할 태스크 부담이 늘어남을 의미한다. 따라서 Fig. 5에 나타나 있는 것처럼 w가 증가할수록 사용자 장치의 평균 CPU 클럭 속도 또한 증가함을 볼 수 있다. 또한 공통적으로 평균 대기열 길이가 증가함에 따라, 즉 V값이 커짐에 따라 목적함수를 감소하려는 방향성이 커지게 되므로 시스템 전력 소비 및 전파 지연을 감소시키기 위해 사용자 장치의 평균 CPU 클럭 속도가 점차 증가함을 확인할 수 있다. 반대로 Fig. 6에서는 w가 커짐에 따라 LEO 위성의 평균 CPU 클럭 속도가 감소하고, 평균 대기열 길이가 증가함에 따라서도 마찬가지로 LEO 위성의 평균 CPU 클럭 속도가 감소함을 볼 수 있다. 앞선 Fig. 4, 5, 6을 통해 제안한 알고리즘의 동작 방향성을 확인할 수 있다. V값을 조절함에 따라 목적함수 변화에 민감한 정도를 제한할 수 있는데, V값을 증가시키면 목적함수 변화에 민감하게 되고 반대로 tradeoff 관계인 대기열 길이 변화에 덜 민감하게 되어 평균 목적함수 값은 감소, 평균 대기열 길이는 증가하게 된다. 이는 평균 대기열 길이를 크게 감소시키고 평균 목적함수 값은 크게 증가시키는 오프로딩의 선택 비율을 감소시키게 되며, 오프로딩을 덜 하게 되니 처리해야 할 태스크들이 사용자들의 장치에 쌓여 사용자 장치의 평균 CPU 클럭 속도의 증가, LEO 위성의 평균 CPU 클럭 속도의 감소를 촉발시키게 된다. 이 모든 변화는 제안한 알고리즘이 취지에 맞게 잘 작동함을 보여주며, 합리적인 범위 내에서 목적함수 값과 대기열 길이의 균형을 잘 조절한다고 볼 수 있다.

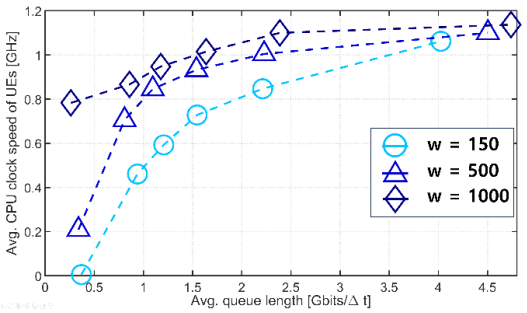

Fig. 5. Avg. queue length - Avg. CPU clock speed of UEs

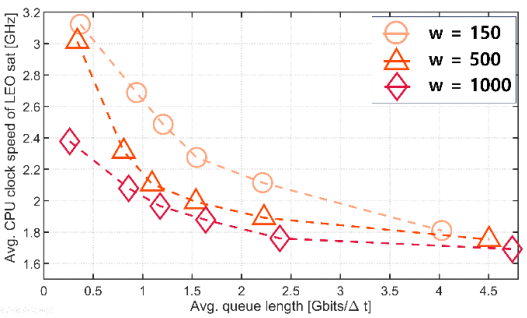

Fig. 6. Avg. queue length - Avg. CPU clock speed of LEO sat

V. 결 론

본 논문에서는 지상-우주 엣지 컴퓨팅 프레임워크를 위해 Lyapunov 최적화를 활용한 동적 오프로딩 및 CPU 스케일링 알고리즘을 제안했다. LEO 위성 네트워크 토폴로지의 특성을 고려한 환경에서 제안한 알고리즘의 시뮬레이션 결과는 다른 비교 알고리즘들에 비해 비슷한 정도의 평균 대기열 길이를 유지하면서 훨씬 적은 평균 전력을 소모하는 것을 보여주었다. 또한 전체적인 결과의 경향성을 포괄적으로 분석해봤을 때, 정의한 목표함수 취지에 알맞게 제안한 알고리즘이 합리적으로 동작함을 확인할 수 있었다. 다만 본 논문에서 제안하는 프레임워크와 알고리즘은 사용자들과 연결된 단 하나의 LEO 위성에서만 프로세싱이 처리되는 것으로 모델링이 진행되었다. 저궤도 위성 네트워크가 폭발적으로 확장되고 있는 만큼 추후에는 요청되는 태스크 오프로딩을 근처의 여러 LEO 위성들 중 목적함수에 따라 선택적으로 수행할 수 있는 프레임워크로 확장할 수 있다고 충분히 예상된다. 우리는 더욱 현실적이고 범용적 연구 진행을 위해 노력할 것이며, 또한 이 기술이 앞으로의 현실적인 저궤도 위성을 활용한 엣지 컴퓨팅의 발전을 촉발하며 더 안정적이고 좋은 성능의 방향으로 연구를 진행하는 데 있어 중요한 역할을 할 것으로 충분히 기대하는 바이다.

부 록

A. Proof of Lemma 1.

Proof: 사용자의 작업량 대기열 모델 (2)과 수학적 사실 $([X]^+)^2 \le X^2$을 통해 다음과 같은 관계를 유도할 수 있다.

$$\begin{aligned} Q_{u,i}^2(t+1) &= \left(Q_{u,i}(t) - \frac{\big(1-\theta_i(t)\big)c_{u,i}(t)}{\gamma} - \theta_i(t)r'(t) + a_i(t) \right)^2 \\ &\le Q_{u,i}(t)^2 \\ &\quad - 2Q_{u,i}(t)\left(\frac{\big(1-\theta_i(t)\big)c_{u,i}(t)}{\gamma} + \theta_i(t)r'(t) - a_i(t) \right) \\ &\quad + \left(\frac{\big(1-\theta_i(t)\big)c_{u,i}(t)}{\gamma} + \theta_i(t)r'(t) - a_i(t) \right)^2 \\ &\le Q_{u,i}(t)^2 \\ &\quad - 2Q_{u,i}(t)\left(\frac{\big(1-\theta_i(t)\big)c_{u,i}(t)}{\gamma} + \theta_i(t)r'(t) - a_i(t) \right) + \frac{c_{u,max}^2}{\gamma} \\ &\quad + \big(r'(t)\big)^2 \\ &\quad + a_{max}^2 \ . \end{aligned} \tag{11}$$

유도한 부등식 (11)을 Lyapunov drift 함수 형태 (8)로 만들기 위해 정렬하면 다음과 같다.

$$\begin{aligned} &Q_{u,i}^2(t+1) - Q_{u,i}^2(t) \\ &\le -2Q_{u,i}(t)\left(\frac{\big(1-\theta_i(t)\big)c_{u,i}(t)}{\gamma} + \theta_i(t)r'(t) - a_i(t) \right) \\ &\quad + \frac{c_{u,max}^2}{\gamma} + \big(r'(t)\big)^2 + a_{max}^2 \ . \end{aligned} \tag{12}$$

일련의 유도 과정 반복을 통해 LEO 위성 처리 대기열 모델 (3)에 대해서도 유사한 결과를 다음과 같이 이끌어낼 수 있다.

$$\begin{aligned} &Q_s^2(t+1) - Q_s^2(t) \\ &\le -2Q_s(t)\left(N_c\frac{c_s(t)}{\gamma} - \sum_{i=1}^{I}\theta_i(t)r'(t) \right) + N_c^2\frac{c_{s,max}^2}{\gamma^2} \\ &\quad + r^2(t) \ . \end{aligned} \tag{13}$$

타임 슬롯 t에 대해, 수식 (10)-(12)와 Lyapunov 함수 (7), Lyapunov drift 함수 (8)을 조합하여 재정의한 목적함수인 Lyapunov drift-plus-penalty 함수 (9)의 upper bound를 얻을 수 있다.

$$\begin{aligned} &\Delta\big(L(t)\big) + V\mathbb{E}\big\{P_{sys}(t) + wL_{sys}(t) \mid \mathbf{Q}(t)\big\} \\ &\le J + V\mathbb{E}\big\{P_{sys}(t) + wL_{sys}(t) \mid \mathbf{Q}(t)\big\} \\ &\quad - \sum_{i=1}^{I}\mathbb{E}\left\{ \left(\frac{\big(1-\theta_i(t)\big)c_{u,i}(t)}{\gamma} + \theta_i(t)r'(t) \right) Q_{u,i}(t) \mid \mathbf{Q}(t) \right\} \\ &\quad - \mathbb{E}\left\{ \frac{N_c c_s(t)}{\gamma} Q_s(t) \mid \mathbf{Q}(t) \right\}. \end{aligned} \tag{14}$$

이 때, 타임 슬롯 t에 우리가 제어하지 않는 항 $J = \frac{1}{2}\left(\sum_{i=1}^{I}\left\{\frac{c_{u,max}^2}{\gamma} + a_{max}^2 + 2\lambda_u Q_{u,i}(t)\right\} + \frac{N_c^2 c_{s,max}^2}{\gamma^2} + 2r^2(t) + 2r(t)Q_s(t)\right)$이다. 이것으로 *Lemma* 1의 증명을 마친다.

References

[1] J. Ryu, et al., "Interference analysis between satellites for LEO satellite constellation networks," *J. KICS*, vol. 47, no. 8, pp. 1051-1054, 2022. (https://doi.org/10.7840/kics.2022.47.8.1051)

[2] J. P. Choi, et al., "Challenges for efficient and seamless space-terrestrial heterogeneous networks," *IEEE Commun. Mag.*, vol. 53, no. 5, pp. 156-62, May 2015.

[3] Y. Mao, et al., "A survey on mobile edge computing: The communication perspective,"

IEEE Commun. Surv. & Tuts., vol. 19, no. 4, pp. 2322-2358, 2017.

[4] N. Cheng, et al., "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas in Commun.*, vol. 37, no. 5, pp. 1117-29, May 2019.

[5] Z. Zhang, et al., "Satellite mobile edge computing: Improving QoS of high-speed satellite-terrestrial networks using edge computing techniques," *IEEE Network*, vol. 33, no. 1, pp. 70-76, Jan. 2019.

[6] C. Qiu, et al., "Deep q-learning aided networking caching and computing resources allocation in software-defined satellite-terrestrial networks," *IEEE Trans. Veh. Technol.*, pp. 1-1, 2019.

[7] X. Cao, et al., "Edge-assisted multi-layer offloading optimization of leo satellite-terrestrial integrated networks," *IEEE J. Sel. Areas in Commun.*, vol. 41, no. 2, pp. 381-398, Feb. 2023.

[8] M. Ra, et al., "Energy-delay tradeoffs in smartphone applications," in *Proc. 8th Int. Conf. Mobile Syst., Appl., and Serv.*, pp. 255-270, San Francisco California, USA, Jun. 2010.

[9] M. Neely, *Stochastic network optimization with application to communication and queueing systems*, Springer Nature, 2022.

[10] J. Kwak, et al., "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.

김 정 환 (Jeong-hwan Kim)

2022년 2월 : 대구경북과학기술원 기초학부 졸업

2022년 3월~현재 : 대구경북과학기술원 전기전자컴퓨터공학과 통합과정

<관심분야> 네트워크, 위성통신, 엣지 컴퓨팅, 동적 최적화

곽 정 호 (Jeong-ho Kwak)

2008년 8월 : 아주대학교 전자공학부 학사

2011년 2월 : 한국과학기술원 전기 및 전자공학과 석사

2015년 2월 : 한국과학기술원전기 및 전자공학과 박사

2015년~2017년 : 캐나다 INRSEMT 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