https://doi.org/10.7840/kics.2024.49.5.718

시/공간적 예산 공유를 활용한 유연한 콘텐츠 캐싱 알고리즘

박 용 문◆, 김 영 진°

Flexible Content Caching Exploiting Spatio-Temporal Budget Sharing

Yongmoon Park◆, Yeongjin Kim°

요 약

미디어 콘텐츠의 소비가 늘어나면서 MEC 서버를 이용하여 콘텐츠를 짧은 지연시간으로 전달할 수 있는 콘텐츠 캐싱 기술이 중요해졌다. 하지만 콘텐츠 캐싱을 위한 MEC 서버의 캐싱 자원은 한정되어있으며 같은 양의 캐싱 자원을 사용하더라도 시간대 혹은 지역에 따라 얻을 수 있는 캐싱 효과는 달라진다. 본 연구에서는 시/공간적으로 캐싱 예산을 유연하게 활용하여 콘텐츠 캐싱의 효과를 극대화하는 *ProCache* 알고리즘을 제안한다. 특히, 콘텐츠별 데이터 크기와 원격 콘텐츠 서버의 위치가 다를 수 있는 환경에서 Lyapunov 최적화를 활용하여 주어진 시간 평균 캐싱 예산 내에서 콘텐츠의 전달시간을 최소화한다. YouTube 데이터셋 기반 시뮬레이션을 이용하여 제안한 *ProCache* 알고리즘의 콘텐츠 요청당 평균 전달시간과 hit ratio를 평가한다. 시뮬레이션 결과를 통하여 *ProCache* 알고리즘이 시/공간적 예산 공유를 통하여 기존 콘텐츠 캐싱 알고리즘의 성능을 상회함을 확인한다.

키워드 : 콘텐츠 캐싱, Lyapunov 최적화, 전달 지연시간 감소, MEC 서버
Key Words : Content caching, Lyapunov optimization, delivery latency reduction, MEC server

ABSTRACT

As media data consumption increases, the content caching technology that delivers content with low latency using an MEC server has been proposed. However, the caching resources of the MEC server are limited, and even if the same amount of caching resources are used, the effectiveness of content caching changes depending on the time zone or region. In this paper, we propose a *ProCache* that maximizes the effectiveness of content caching by using caching resources flexibly in spatial and temporal domains. In particular, when each content has a different size and location of remote content server, *ProCache* minimizes content delivery latency under a given time-average caching budget using Lyapunov optimization. Based on the simulation using the YouTube dataset, we evaluate the content delivery time per request and the cache hit ratio of the *ProCache*. The simulation results show that the proposed *ProCache* outperforms existing content caching algorithms by enabling spatio-temporal budget sharing.

※ 본 연구는 2024년도 정보통신기획평가원 (No.RS-2022-00155915 인공지능융합혁신인재양성(인하대학교), 2021-0-02201 사용자 프라이버시를 보존하는 비디오 캐싱을 위한 연합 학습 시스템, 2022-0-00448 인간처럼 회상이 가능한 인공 신경망 지속학습 플랫폼 개발), 및 한국연구재단 (No. RS-2023-00240019)의 지원을 받아 수행된 연구임.
◆ First Author : Inha University Department of Electrical and Computing Engineering, sky123pq@inha.edu, 학생회원
° Corresponding Author : Inha University Department of Electrical and Computing Engineering, yj.kim@inha.ac.kr, 정회원
논문번호 : 202401-008-C-RN, Received January 3, 2024; Revised January 21, 2024; Accepted January 21, 2024

Ⅰ. 서 론

최근 SNS와 비디오 서비스로 인한 모바일 비디오 데이터 수요가 급증하고 있다. 일반적으로, 콘텐츠 서비스 사업자는 사용자에게 콘텐츠 서비스를 제공할 시 콘텐츠 캐싱 기술을 활용한다. 콘텐츠 캐싱이란 사용자와 가까이 위치한 모바일 엣지 컴퓨팅(MEC) 서버에 특정 콘텐츠를 미리 저장하고, 해당 콘텐츠 요청이 발생했을 때 저장한 콘텐츠를 사용자에게 전달하는 기술이다. MEC 서버로부터 콘텐츠를 전달받는 경우, 백홀 네트워크에 의존하지 않기 때문에, 원격 콘텐츠 서버에서 전달받는 경우보다 더 낮은 전달 지연시간을 달성할 수 있다. 그러나 MEC 서버의 캐싱 자원은 한정적이기 때문에 적은 수의 콘텐츠만 캐싱할 수 있으며, 콘텐츠 서비스 사업자는 가장 높은 캐싱 효과를 얻을 수 있는 콘텐츠를 선택 및 캐싱해야 한다.

일반적으로, 최대한 낮은 지연시간으로 콘텐츠를 전달하기 위해서는 사용자들로부터 빈번한 요청이 발생하는 콘텐츠를 위주로 캐싱해야 한다[1]. 기존 콘텐츠 캐싱 연구에서는 시간에 대하여 변하지 않는 콘텐츠의 인기도를 가정하거나, 특정 확률 분포의 콘텐츠의 인기도를 가정한 상황에서 최적화 기법을 적용하여 콘텐츠의 캐싱을 결정하였다[2-5]. 그러나 콘텐츠의 인기도는 지역에 따라 다를 수 있으므로 지역별 MEC 서버의 캐싱 전략은 달라야 한다. 이와 관련하여, 계층적 캐시 구조를 포함한 다중 지역 서비스를 위한 콘텐츠 캐싱 연구가 진행되었다[6-9]. 또한, 콘텐츠 서비스 사업자가 사용할 수 있는 캐싱 자원(예: 에너지, 스토리지, 예산)은 시간 평균적으로 한정되어있기 때문에, 시간 축에서도 자원 효율적으로 콘텐츠 캐싱이 제어될 수 있어야 한다[10,11].

콘텐츠를 캐싱했을 때 얻을 수 있는 전체 서비스의 지연시간 감소 효과는 캐싱한 콘텐츠의 인기도뿐만 아니라 콘텐츠의 원본이 저장되어있는 원격 콘텐츠 서버의 위치와도 관련이 있다. 콘텐츠를 요청하는 사용자가 위치한 지역이 해당 콘텐츠의 원격 콘텐츠 서버와 멀리 떨어져 있을수록 캐싱을 통해 얻을 수 있는 지연시간 감소 효과가 크다.

본 연구에서는 각 콘텐츠의 예측 조회수가 [14]로부터 주어졌을 때, 여러 지역의 MEC 서버에서의 콘텐츠 캐싱 제어를 통하여 전체 사용자의 평균 콘텐츠 전달 지연시간을 최소화한다. 특히, 시간 평균적으로 콘텐츠 서비스 사업자가 사용 가능한 캐싱 예산이 주어지는 상황을 고려하고, 콘텐츠별 크기와 원격 콘텐츠 서버의 위치가 다를 수 있는 상황을 고려한다. 시간에 따라 지역별 캐싱할 콘텐츠를 제어하여 정의한 문제를 해결하는 *ProCache* 알고리즘을 제안한다. 제안한 알고리즘은 가상 대기열 기반 Lyapunov 최적화 기법과 Knapsack 문제를 활용하여 캐싱 효과가 높은 지역과 시간대에 집중하여 캐싱 예산을 투자함으로써 기존 연구보다 유연한 캐싱 예산 사용을 통해 콘텐츠 전달 지연시간을 효과적으로 감소시킬 수 있다. 또한, 지역 간에 캐싱 예산 관점에서 엮인 상황임에도 불구하고 *Procache* 알고리즘은 지역별로 분산화되어 동작한다.

시뮬레이션 기반 성능 평가에서는 실제 콘텐츠 요청 수 데이터셋과 네트워크 성능 트레이스를 활용하여 *ProCache* 알고리즘이 시/공간적으로 유연하게 캐싱 예산을 사용하여 시간 평균 콘텐츠 전달 지연시간을 최소화함을 확인한다.

Ⅱ. 시스템 모델

2.1 서비스 모델

그림 1은 콘텐츠의 원본이 저장되어있는 원격 콘텐츠 서버와 제한된 양의 콘텐츠를 캐시 할 수 있는 사용자와 가깝게 위치한 MEC 서버가 존재하는 콘텐츠 캐싱 시스템구조를 나타낸다. 각 MEC 서버 $e \in \boldsymbol{E} \equiv \{1,...,E\}$는 지역별 하나씩 설치되어있고 인터넷을 통해 회계 서버와 원격 콘텐츠 서버와 통신 가능하다. 콘텐츠 서비스 사업자는 MEC 서버에 콘텐츠를 캐시 할 때 MEC 사업자 측에 사용한 캐싱 용량에 비례하여 시간당 캐싱 요금을 지급한다. 시스템은 이산 시간 시스템이며 각 타임 슬롯은 $t \in \boldsymbol{T} = \{0,1,....,T\}$ 와 같이 정의된다. 각 타임 슬롯의 시작 시점에서 캐싱 콘텐츠 목록을 갱신한다.

2.2 콘텐츠 및 캐싱 모델

각 콘텐츠 $c \in \boldsymbol{C}$ 는 크기 s_c(bits)를 가지며 원격 콘텐츠 서버 중 한 곳에만 저장되어있다. 타임 슬롯 t동안 지역 e에 들어오는 콘텐츠 c에 대한 요청 수를 $a_c^e(t)$로 정의하며, 해당 타임 슬롯이 시작되는 시점에 주어진다고 가정한다. 지역 e의 사용자가 콘텐츠를 요청하면, MEC 서버 e에 해당 콘텐츠가 캐시되어있는 경우 MEC 서버에서 사용자에게 콘텐츠를 직접 전달하고 이를 캐시 적중(cache hit)이라 한다. 반대로, 요청한 콘텐츠가 MEC 서버 e에 저장되어있지 않은 경우, 사용자는 MEC 서버보다 멀리 위치한 원격 콘텐츠 서버로부터 직접 콘텐츠를 받아오고 이를 캐시 미스(cache miss)라고 한다. 타임 슬롯 t에서의 콘텐츠 캐싱 결정은 변수

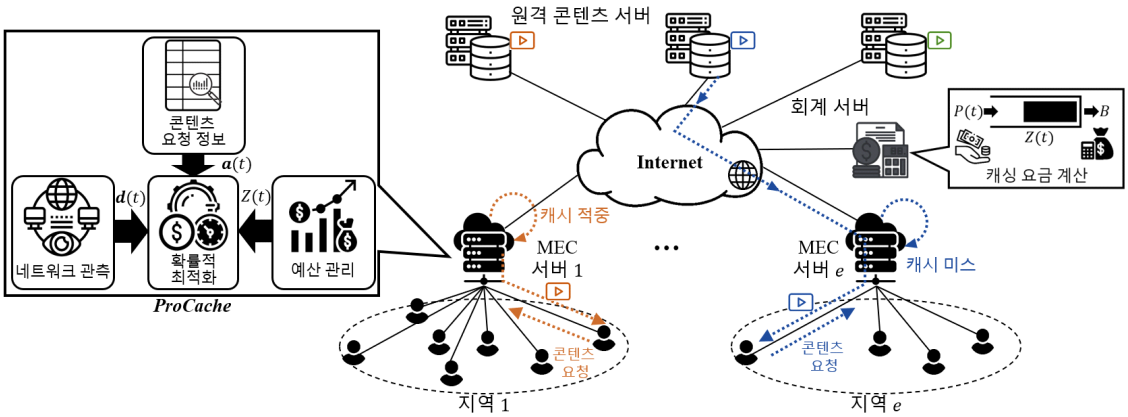


그림 1. 사전 콘텐츠 캐싱 시스템 전체 구조
Fig. 1. Overall architecture of a proactive content caching system

$\theta_c^e(t) \in \{0,1\}$로 나타내며 $\theta_c^e(t)=1$은 콘텐츠 c를 MEC 서버 e에 캐싱함을 의미하고 $\theta_c^e(t)=0$는 캐싱하지 않음을 의미한다. 각 MEC 서버가 콘텐츠를 캐싱할 수 있는 저장소 용량을 O^e(bits)로 표현하며 다음과 같은 제약사항이 존재한다.

$$\Sigma_{c\in\boldsymbol{C}}\, s_c\theta_c^e(t) \le O^e,\ \forall e\in\boldsymbol{E}, t\in\boldsymbol{T}. \tag{1}$$

2.3 캐싱 비용 모델

각 MEC 서버 e의 캐시 사용에 대하여 종량제 비용 모델을 고려한다. 단위 시간 동안의 단위 용량의 캐시 저장소를 사용 경우에 지급해야 하는 비용을 p^e($/bit)로 정의한다. 모든 콘텐츠의 캐싱 결정 변수를 $\boldsymbol{\theta}(t)=(\theta_c^e(t), \forall c\in\boldsymbol{C}, e\in\boldsymbol{E})$로 나타낼 때, 타임 슬롯 t에서 모든 지역에 대해 지불해야 하는 콘텐츠 캐싱 비용 $P(t)$는 다음과 같다.

$$P(t)=\Sigma_{e\in\boldsymbol{E}}\Sigma_{c\in\boldsymbol{C}}\, s_c\theta_c^e(t)p^e. \tag{2}$$

또한, 콘텐츠를 캐싱하는데 지불하는 비용은 모든 지역에 대하여 시간 평균적으로 예산 B($) 이하로만 가능하며 다음과 같은 제약사항이 존재한다.

$$\lim_{T\to\infty}\frac{1}{T}\Sigma_{t\in\boldsymbol{T}}\,\mathbb{E}\{P(t)\}\le B. \tag{3}$$

회계 서버는 모든 지역의 캐싱 사용 비용을 취합하고, 매 타임 슬롯 t의 시작 시점에 각 MEC 서버로 가용 캐싱 예산 정보를 공유한다.

2.4 네트워크 모델

타임 슬롯 t에서, MEC 서버 e가 사용자에게 캐시된 콘텐츠를 전송하는 경우의 전송 속도는 $d^e(t)$ (bits/second)로 정의한다. 반대로, 콘텐츠 c를 저장한 원격 콘텐츠 서버에서 사용자에게 콘텐츠 c를 직접 전송하는 경우의 전송 속도는 $d_c^r(t)$로 정의한다. 일반적으로, 원격 콘텐츠 서버는 MEC 서버보다 사용자에게 훨씬 더 멀리 위치하기 때문에 $d_c^r(t)\le d^e(t)$ 관계가 성립한다. 위의 정의에 따라 MEC 서버가 사용자에게 캐싱 된 콘텐츠 c를 전달시 발생하는 지연시간은 $l_c^e(t)=s_c/d^e(t)$ (seconds)가 되며, 원격 콘텐츠 서버가 사용자에게 콘텐츠를 직접 전송시 발생하는 지연시간은 $l_c^r(t)=s_c/d_c^r(t)$ (seconds)가 된다. 마지막으로, 타임 슬롯 t에서 주어지는 모든 네트워크 상태를 $\boldsymbol{d}(t)=(d^e(t), \forall e\in\boldsymbol{E}, d_c^r(t), \forall c\in\boldsymbol{C})$로 나타낸다.

그림 1에서 나타낸 것과 같이 회계 서버로부터 전달받은 가용 캐싱 예산 정보와 네트워크 모니터링 정보, 콘텐츠 요청 수 정보를 바탕으로 각 MEC 서버는 해당 지역의 캐싱 결정 $\boldsymbol{\theta}^e(t)$를 갱신한다.

Ⅲ. 문제 정의 및 제안 기법

3.1 문제 정의

본 절에서는 사용자의 콘텐츠 전달 지연시간을 최소화하는 문제를 정의하고 이를 해결하기 위한 *ProCache* 알고리즘을 살펴본다. 먼저 타임 슬롯 t에서 발생하는 모든 콘텐츠 요청의 전달 지연시간 $L(t)$는 다음과 같이 계산된다.

$$L(t)=\Sigma_{e\in\boldsymbol{E}}\Sigma_{c\in\boldsymbol{C}}a_c^e(t)\left\{\theta_c^e(t)l_c^e(t)+(1-\theta_c^e(t))l_c^r(t)\right\}. \tag{4}$$

이러한 상황에서 $\boldsymbol{\theta}=(\theta_c^e(t),\forall c\in\boldsymbol{C},e\in\boldsymbol{E},t\in\boldsymbol{T})$를 제어하여 시간 평균 관점에서 모든 콘텐츠 요청의 전달 지연시간을 최소화하는 문제 (P1)을 다음과 같이 정의한다.

$$(\mathbf{P1}):\min_{\boldsymbol{\theta}}\lim_{T\to\infty}\frac{1}{T}\Sigma_{t\in\boldsymbol{T}}\mathbb{E}\{L(t)\},$$
$$s.t.\begin{cases}(\mathbf{C1}):\Sigma_{c\in\boldsymbol{C}}s_c\theta_c^e(t)\le O,\ \forall e\in\boldsymbol{E},t\in\boldsymbol{T},\\(\mathbf{C2}):\theta_c^e(t)\in\{0,1\},\qquad\forall e\in\boldsymbol{E},t\in\boldsymbol{T},c\in\boldsymbol{C},\\(\mathbf{C3}):\lim_{T\to\infty}\frac{1}{T}\Sigma_{t\in\boldsymbol{T}}\mathbb{E}\{P(t)\}\le B.\end{cases}$$

여기서, (C1)은 MEC 서버별 캐싱 용량에 대한 제약사항, (C2)는 캐싱 결정에 대한 제약사항, (C3)은 시간 평균 가용 가능한 캐싱 예산에 대한 제약사항이다.

3.2 알고리즘 유도

가상 대기열 기반의 Lyapunov 최적화를 사용하여[12] 시간 평균 최소화 문제 (P1)을 매 타임 슬롯마다 정의되는 문제로 변형한다. 먼저 캐싱 예산과 관련된 가상 대기열 $Z(t)$를 다음과 같이 정의한다. 단 $Z(0)<\infty$이다.

$$Z(t+1)=\max(Z(t)+P(t)-B,0). \tag{5}$$

가상 대기열을 안정화한다면 (C3)를 만족시킬 수 있다. 즉, 가상 대기열을 안정화하는 것으로 전체 시간 평균적으로 캐싱 비용 지불에 대한 제약사항을 만족시킬 수 있으며 $Z(t)$를 통하여 주어진 예산에 비해 얼마나 캐싱 예산을 사용하고 있는지 파악할 수 있다. 이후 $Z(t)$에 기반하여 Lyapunov function $\mathbb{L}(Z(t))$와 Lyapunov drift $\Delta\mathbb{L}(Z(t))$를 다음과 같이 정의한다.

$$\mathbb{L}(Z(t))=\frac{1}{2}Z(t)^2. \tag{6}$$

$$\Delta\mathbb{L}(Z(t))=\mathbb{E}\{\mathbb{L}(Z(t+1))-\mathbb{L}(Z(t))|Z(t)\}. \tag{7}$$

다음으로 Lyapunov drift와 V가 곱해진 타임 슬롯 t에서의 모든 콘텐츠 요청의 전달 지연시간, 두 개의 항의 합인 Lyapunov drift-plus-penalty (DPP)를 다음과 같이 정의한다.

$$\Delta\mathbb{L}(Z(t))+V\mathbb{E}\{L(t)|Z(t)\}. \tag{8}$$

이때 V는 콘텐츠 전달 지연시간의 최소화를 달성하려는 힘과 캐싱 비용을 예산을 초과하지 않도록 하려는 힘 사이를 조절하는 양수의 trade-off 파라미터이다. DPP의 상한은 다음과 같이 정의된다.

$$\begin{aligned}&\Delta\mathbb{L}(Z(t))+V\mathbb{E}\{L(t)|Z(t)\}\\&\le J+V\mathbb{E}\{L(t)|Z(t)\}+\mathbb{E}\{Z(t)(P(t)-B)|Z(t)\}.\end{aligned} \tag{9}$$

이때 $J=0.5((P^{\max})^2+B^2)$이며 $P^{\max}$는 $P(t)$의 상한을 의미한다.

DPP의 Lyapunov drift는 관측 가능한 정보로 계산할 수 없으므로 DPP의 상한을 최소화한다. DPP의 상한에서 $\boldsymbol{\theta}(t)$와 관련 없는 항을 제거하면 다음과 같은 최소화 문제로 정리할 수 있다.

$$\begin{aligned}&\min_{\boldsymbol{\theta}(t)}VL(t)+Z(t)P(t)\\&=\Sigma_{e\in\boldsymbol{E}}\Sigma_{c\in\boldsymbol{C}}s_c\theta_c^e(t)\left\{V\left(\frac{a_c^e(t)}{d^e(t)}-\frac{a_c^e(t)}{d_c^r(t)}\right)+Z(t)p^e\right\}\\&+\Sigma_{e\in\boldsymbol{E}}\Sigma_{c\in\boldsymbol{C}}V\frac{a_c^e(t)s_c}{d_c^r(t)}\\&\qquad s.t.\ (\mathbf{C1}),(\mathbf{C2}).\end{aligned} \tag{10}$$

다음으로, $\boldsymbol{\theta}(t)$와 관련 없는 $Va_c^e(t)s_c/d_c^r(t)$ 항을 제거하고 지역 단위로 문제를 쪼개면 다음과 같은 최대화 문제로 정리할 수 있다. 각 지역 e에 대해서,

$$\begin{aligned}&\max_{\boldsymbol{\theta}^e(t)}\Sigma_{c\in\boldsymbol{C}}s_c\theta_c^e(t)\left\{V\left(\frac{a_c^e(t)}{d_c^r(t)}-\frac{a_c^e(t)}{d^e(t)}\right)-Z(t)p^e\right\}.\\&\qquad s.t.\ (\mathbf{C1}),(\mathbf{C2}).\end{aligned} \tag{11}$$

이때, 표현의 간소화를 위하여 $\omega_c^e(t)$를 $V(a_c^e(t)/d_c^r(t)\ -a_c^e(t)/d^e(t))-Z(t)p^e$로 정의하면, 다음과 같이 문제를 정리할 수 있다.

$$\begin{aligned}&(\mathbf{P2}):\max_{\boldsymbol{\theta}^e(t)}\Sigma_{c\in\boldsymbol{C}_+^e}(t)\ s_c\theta_c^e(t)\omega_c^e(t)\\&\qquad s.t.\ (\mathbf{C1}),(\mathbf{C2}).\end{aligned}$$

여기서 $\boldsymbol{C}_+^e(t)$는 $\omega_c^e(t)>0$를 만족하는 콘텐츠의 집합이다. $\omega_c^e(t)\le 0$를 만족하는 콘텐츠를 캐싱하는 경우 (P2)의 목적함수 값을 감소시키기 때문에 캐싱하지 않

는다. 따라서, *ProCache* 알고리즘은 콘텐츠 집합 $C_+^e(t)$ 내에서 캐싱할 콘텐츠를 선택한다.

3.3 알고리즘 설명

(P2) 문제는 0-1 Knapsack 문제로 표현될 수 있으며, 이는 NP-hard 문제로 알려져 있으므로 다항 시간 내에 최적의 $\theta_e(t)$을 찾을 수 없다. 그러므로 낮은 복잡도를 가지면서 0.5 approximation ratio를 보장하는 greedy 기반의 알고리즘을 제안한다[13]. *ProCache*는 다음 두 개의 greedy 알고리즘들로 결정된 $\theta_e(t)$중 하나를 선택한다.

Value-first greedy: $s_c\omega_c^e(t)$ 값이 큰 순으로 콘텐츠 $c \in C_+^e(t)$를 캐싱한다. 즉, 콘텐츠의 크기 s_c가 크면서, 원격 콘텐츠 서버의 전송 속도 $d_c^r(t)$와 MEC 서버의 전송 속도 $d^e(t)$의 차이가 크거나 해당 지역 e에서의 콘텐츠 요청 수 $a_c^e(t)$가 높은 콘텐츠를 우선하여 캐싱 콘텐츠를 선정된다. 이 과정은 캐싱한 콘텐츠 크기의 합이 MEC 서버의 캐시 저장소 용량 O^e를 초과하기 전까지 계속 반복된다.

Density-first greedy: $\omega_c^e(t)$ 값이 큰 순으로 콘텐츠 $c \in C_+^e(t)$를 캐싱한다. 즉, Value-first greedy와 동일하게 동작하지만, 콘텐츠를 선정하는 기준에서 콘텐츠의 크기를 고려하지 않는다.

$C_+^e(t)$ 콘텐츠 집합에 대해 두 greedy 알고리즘으로 캐싱할 콘텐츠를 선정한 후 *ProCache*는 두 솔루션 중 (P2)의 목적함수 값이 더 큰 솔루션을 최종 선택한다. 이 과정은 매 타임 슬롯 t마다 반복된다.

Ⅳ. 시뮬레이션 기반 평가

4.1 데이터셋

본 시뮬레이션에서는 2018년 5월 동안 수집된 Youtube 콘텐츠들의 요청 히스토리 데이터셋를 통해 예측한 향후 콘텐츠 요청 수 정보를 사용한다[14]. 총 1574개 콘텐츠를 고려하며, 시뮬레이션 내의 타임 슬롯의 단위는 3시간으로 설정한다. 데이터셋에 기록된 콘텐츠별 duration (seconds) 정보를 기반으로 해상도를 Full High Definition (FHD)로 가정하여 각 콘텐츠의 크기를 설정한다. 이때, 콘텐츠의 평균 크기는 약 230MB이며, 최소 2MB, 최대 1188MB의 범위를 갖는다. 각 MEC 서버의 전송 속도 $d^e(t)$와 각 원격 콘텐츠 서버의 전송 속도 $d_c^r(t)$는 실제 네트워크 트레이스를 기반으로 한다[6]. 이때, MEC 서버의 평균 전송 속도는 39.64Mbps이고 원격 콘텐츠 서버의 평균 전송 속도는 11.40Mbps이다.

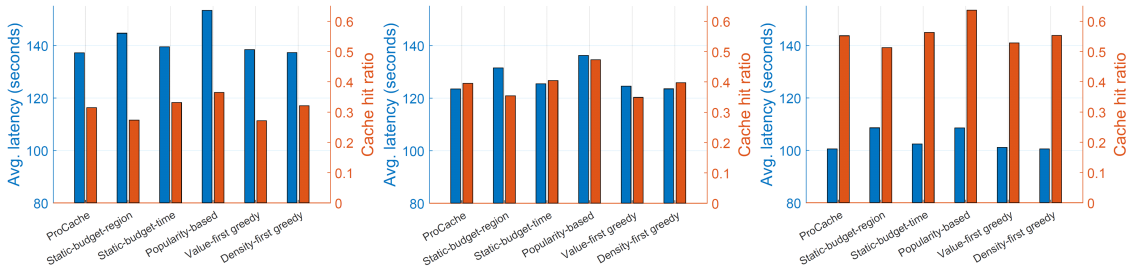
4.2 시뮬레이션 세팅

시뮬레이션은 3개의 지역에서 콘텐츠 캐싱을 서비스하는 상황을 가정하며 각 지역의 인구수의 비율은 약 1:2:4로 설정한다. 콘텐츠의 원본이 저장되어있는 원격 콘텐츠 서버는 총 5개이며, 각 원격 콘텐츠 서버의 전송 속도의 비율은 약 5:4:3:2:1로 설정 한다. 시뮬레이션은 각각 다른 조합의 MEC 서버의 콘텐츠 캐싱 저장소 용량 O^e와 시간 평균 예산 B를 설정한다.

4.3 성능 평가

본 시뮬레이션에서는 콘텐츠 요청당 평균 콘텐츠 전달 지연시간을 성능 지표로 삼는다. 제안한 *ProCache*과 성능을 비교할 콘텐츠 전달 지연시간을 최소화하는 캐싱 알고리즘들은 다음과 같다. 1) Static-budget-region: 지역별로 균등하게 시간 평균 예산을 $B/3$ 씩 할당한다. 해당 알고리즘과 *ProCache*의 성능을 비교하여 지역 간 유연하게 캐싱 예산을 사용할 때 얻을 수 있는 효과를 확인한다. 2) Static-budget-time: 시간대마다 고정된 예산 B를 사용하여 모든 지역에서의 캐싱 콘텐츠를 결정한다. 이 경우 각 시간대에서 사용할 수 있는 예산은 B를 넘지 못한다. 해당 알고리즘과 *ProCache*의 성능을 비교하여 시간 축에서 유연하게 캐싱 예산을 사용할 때 얻을 수 있는 효과를 확인한다. 3) Popularity-based: 요청 수가 높은 콘텐츠 순으로 캐싱 콘텐츠를 결정한다. 해당 알고리즘과 *ProCache*의 성능을 비교하여 각 MEC 서버와 원격 콘텐츠 서버의 전송 속도를 고려하여 캐싱 콘텐츠를 결정하는 경우 얻을 수 있는 효과를 확인한다. 4) Value-first greedy: $s_c\omega_c^e(t)$가 가장 높은 콘텐츠 순으로 캐싱 콘텐츠를 결정한다. 5) Density-first greedy: $\omega_c^e(t)$가 가장 높은 콘텐츠 순으로 캐싱 콘텐츠를 결정한다. 4)와 5)의 알고리즘과 *ProCache*의 성능을 비교하여 4)와 5)를 앙상블 했을 때 얻을 수 있는 효과를 확인한다.

그림 2에서는 각 캐싱 알고리즘의 콘텐츠 요청당 평균 콘텐츠 전달 지연시간과 캐시 hit ratio를 비교한다. 그림 2(a), (b), (c)에서 캐싱 예산과 캐싱 용량은 캐싱 예산과 MEC 서버의 캐시 저장 용량이 전체 콘텐츠의 크기 합 대비 얼마나 주어지는지를 의미한다. 그림 2에서 Popularity-based 알고리즘의 경우, 콘텐츠의 요청 수를 기준으로 캐싱 콘텐츠를 결정하기 때문에 hit ratio가 가장 높지만, 콘텐츠 전달 지연시간이 가장 높게 측

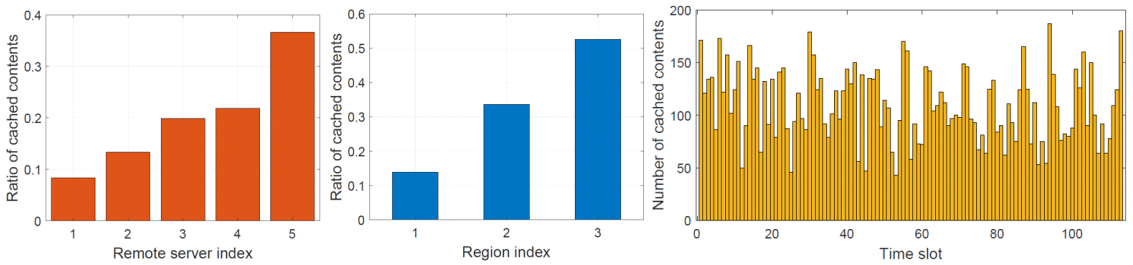


(a) 캐싱 예산: 0.7% 캐싱 용량: 1.4% (b) 캐싱 예산: 1.0% 캐싱 용량: 2.1% (c) 캐싱 예산: 2.0% 캐싱 용량: 4.1%

그림 2. 캐싱 예산과 캐시 저장소 용량에 따른 캐싱 알고리즘들의 콘텐츠 요청당 평균 전달 지연시간과 캐시 적중률
Fig. 2. Average delivery latency and cache hit ratio of caching methods for different amounts of caching budget and caching storage capacity relative to total content volume

정된 것을 확인할 수 있다. 이는 콘텐츠 인기도뿐만 아니라 콘텐츠 캐싱을 통한 콘텐츠 전달 지연시간 감소 효과를 고려해야함을 의미한다. 이에 반해, *ProCache*는 콘텐츠 인기도와 콘텐츠 캐싱으로 인한 전달 지연시간 감소를 함께 고려하며, 시/공간적으로 유연하게 캐싱 예산을 사용할 수 있다. 이로 통해 *ProCache*가 가장 낮은 평균 콘텐츠 전달 지연시간을 달성하는 것을 확인할 수 있다. 또한, Density-first greedy와 Value-first greedy를 비교해보았을 때, 둘 중 Density-first greedy가 더 *ProCache*에 가까운 성능을 보인다. 이는 *ProCache*에서 두 greedy 알고리즘을 앙상블 할 때, Density-first greedy 알고리즘이 선택되는 경우가 빈번하다는 것을 의미한다. 마지막으로, 가용 캐싱 예산이 0.7%, 1.0%, 2.0%로 증가하고 MEC 서버의 콘텐츠 캐시 저장 용량이 1.4%, 2.1%, 4.1%로 증가하는 경우, *ProCache*의 콘텐츠 전달 지연시간이 점점 137.2초, 123.5초, 100.5초로 감소하는 것을 확인할 수 있다.

그림 3은 가용 캐싱 예산이 2.0%이고 캐싱 저장 용량이 4.1% 일 때 *ProCache* 알고리즘 동작의 시/공간적인 특성을 보여준다. 그림 3(a)는 각 원격 콘텐츠 서버에 저장된 콘텐츠들의 캐시 되는 비율을 나타낸다. Remote 5의 경우 원격 콘텐츠 서버 중 가장 느린 전송 속도를 가지기 때문에 다른 원격 콘텐츠 서버보다 콘텐츠가 캐시 되는 비율이 더욱 높은 것을 알 수 있다. 이는 *ProCache*가 캐싱할 콘텐츠를 결정할 때 요청 수뿐만 아니라 캐싱을 통해 얻을 수 있는 전달 지연시간 감소 효과를 함께 고려하기 때문이다. 그림 3(b)에서는 캐시된 전체 콘텐츠 중 각 지역의 MEC 서버에 캐시된 콘텐츠의 비율을 나타내며, 지역별 13.8% 33.6% 52.6%의 값을 가진다. 이는 지역별 인구수의 비율이 약 1:2:4였고 *ProCache*가 요청 수의 규모가 큰 지역에 더 많은 캐싱 예산을 할당하기 때문이다. 이를 통해, 지역 간 캐싱 예산 공유의 중요성을 확인할 수 있다. 그림 3(c)는 시간에 따른 모든 지역에서 캐싱된 콘텐츠의 수의 변화를 나타낸 것이다. 시간대에 따라 캐싱된 콘텐츠의 수가 동적으로 변하는 것을 확인할 수 있으며, 주로 저녁 시간대에 많은 수의 콘텐츠를 캐싱하고 새벽 시간대에 적은 수의 콘텐츠를 캐싱하는 것을 확인할 수 있다. 이를 통해 *ProCache*가 캐싱 효과가 낮은 콘텐츠 수요가 낮은 시간대에서는 캐싱 예산을 절약하고 캐싱 효과가 높은 콘텐츠 수요가 높은 시간대에서는 캐싱 예산을 공격적으로 투자하는 시간 축에서의 캐싱 예산 공유의



(a) 원격 콘텐츠 서버별 콘텐츠의 캐싱 선택 비율 (b) MEC 서버별 콘텐츠의 캐싱 선택 비율 (c) 시간에 따른 모든 지역의 캐싱된 콘텐츠 수

그림 3. *ProCache*의 알고리즘 동작의 시간적, 공간적인 특성
Fig. 3. Behavior of *ProCache* in terms of spatio-temporal domain

중요성을 확인할 수 있다.

V. 결 론

본 연구에서는 한정된 캐싱 예산과 관련한 시간 평균 제약조건을 만족하면서 전체 사용자의 시간 평균 콘텐츠 전달 지연시간을 최소화하는 *ProCache* 알고리즘을 제안하였다. 제안한 알고리즘은 지역적, 시간적 상황을 고려하여 콘텐츠의 캐싱 여부를 결정한다. 시뮬레이션을 통하여 기존 콘텐츠 캐싱 알고리즘과의 성능 비교를 통해, 제안한 *ProCache* 알고리즘이 시간 평균 콘텐츠 전달 지연시간을 최소화할 수 있음을 확인하였다. 또한, *ProCache* 알고리즘이 실제로 시/공간적 상황에 따라 얼마나 유연하게 동작하는지 확인하였다.

References

[1] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Commun. Surv. & Tuts.*, vol. 20, no. 4, pp. 3098-3130, May 2018. (https://doi.org/10.1109/COMST.2018.2841349)

[2] Z. Zhao, M. Peng, Z. Ding, W. Wang, and H. V. Poor, "Cluster content caching: An energy-efficient approach to improve quality of service in cloud radio access networks," *IEEE J. Sel. Areas in Commun.,* vol. 34, no. 5, pp. 1207-1221, May 2016. (https://doi.org/10.1109/JSAC.2016.2545384)

[3] T. X. Tran, A. Hajisami, and D. Pompili, "Cooperative hierarchical caching in 5g cloud radio access networks," *IEEE Network*, vol. 31, no. 4, pp. 35-41, Jul. 2017. (https://doi.org/10.1109/MNET.2017.1600307)

[4] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402-8413, Dec. 2013. (https://doi.org/10.1109/TIT.2013.2281606)

[5] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82-89, Aug. 2014. (https://doi.org/10.1109/MCOM.2014.6871674)

[6] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Hybrid content caching in 5g wireless networks: Cloud versus edge caching," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3030-3045, May 2018. (https://doi.org/10.1109/twc.2018.2805893)

[7] A. Gharaibeh, A. Khreishah, and I. Khalil, "An O(1)-competitive online caching algorithm for content centric networking," in *Proc. IEEE INFOCOM*, pp. 1-9, San Francisco, CA, USA, Apr. 2016. (https://doi.org/10.1109/infocom.2016.7524444)

[8] Z. Zheng, L. Song, Z. Han, G. Y. Li, and H. V. Poor, "A stackelberg game approach to proactive caching in large-scale mobile edge networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5198-5211, Aug. 2018. (https://doi.org/10.1109/TWC.2018.2839111)

[9] M. Javedankherad, Z. Zeinalpour-Yazdi, and F. Ashtiani, "Mobility-aware content caching using graph-coloring," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 5666-5670, May 2022. (https://doi.org/10.1109/TVT.2022.3156528)

[10] Y. Zhang, C. Li, T. H. Luan, Y. Fu, W. Shi, and L. Zhu, "A mobility-aware vehicular caching scheme in content centric networks: Model and optimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3100-3112, Apr. 2019. (https://doi.org/10.1109/TVT.2019.2899923)

[11] N. Abedini and S. Shakkottai, "Content caching and scheduling in wireless networks with elastic and inelastic traffic," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 864-874, Jun. 2014. (https://doi.org/10.1109/TNET.2013.2261542)

[12] M. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Commun. Netw.*, pp. 1-211, 2010. (https://doi.org/10.2200/s00271ed1v01y201006

cnt007)

[13] S. Martello and P. Toth, "*Knapsack problems: Algorithms and computer implementations*," John Wiley & Sons, Inc., 1990. (https://doi.org/10.1016/0377-2217(91)90159-s)

[14] Y. M. Park and Y. G. Kim, "Deep learning-based view count prediction for content caching services," *J. KICS*, vol. 48, no. 12, pp. 1559-1567, Dec. 2023. (https://doi.org/10.7840/kics.2023.48.12.1559)

박 용 문 (Yongmoon Park)

2022년 2월 : 인하대학교 전자공학과 졸업

2022년 3월~현재 : 인하대학교 전기컴퓨터공학과 석사 과정

<관심분야> 콘텐츠 캐싱 및 엣지 컴퓨팅

[ORCID:0009-0004-2607-9260]

김 영 진 (Yeongjin Kim)

2011년 2월 : 한국과학기술원 전자공학과

2013년 2월 : 한국과학기술원 전자공학과 석사

2018년 2월 : 한국과학기술원 전자공학과 박사

2018년 3월~2020년 2월 : 삼성전자 senior engineer

2020년 3월~현재 : 인하대학교 전자공학과 조교수

<관심분야> 모바일, 엣지, 클라우드 컴퓨팅

[ORCID:0000-0003-4482-2287]