The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 23, No. 5, pp.9-13, Oct. 31, 2023. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2023.23.5.9
JIIBC 2023-5-2

# 멀티미디어 스트리밍 환경을 위한 캐쉬 성능평가 모델 설계 및 최적성 분석

# Design and Optimality Analysis of Cache Performance Model for Multimedia Streaming Environments

**반효경*, 조경운****

Hyokyung Bahn*, Kyungwoon Cho**

**요 약** 멀티미디어 스트리밍 데이터는 용량이 매우 크고 순차적으로 접근이 이루어지는 특성이 있어 전통적인 캐쉬 환경에서 입출력의 성능을 개선하기 위해 널리 사용되고 있는 LRU 알고리즘이 효과적이지 못한 것으로 알려져 있다. 이에 대한 실험적인 분석은 인터벌 기반 캐쉬의 LRU 대비 성능 상의 비교 우위를 통해 입증된 바 있으나, 이론적인 근거가 증명되지는 않았다. 본 논문에서는 멀티미디어 스트리밍 환경을 위한 캐쉬의 최적성을 분석하기 위해 캐쉬 성능 평가 모델을 설계하고, 이론적으로 최적인 캐쉬 알고리즘을 인터벌 캐쉬에 기반해서 설계한다. 그런 다음 설계된 알고리즘이 스트리밍 데이터의 캐쉬 미스를 최소화하는 교체 알고리즘임을 제안된 모델에 근거한 최적성 분석을 통해 입증한다.

**Abstract** Multimedia streaming data is very large in size and accessed sequentially, making the LRU(Least Recently Used) algorithm widely used to improve I/O performance in traditional caching environments ineffective. Experimental analysis of this has shown the superiority of interval-based caching over LRU, but the theoretical basis has not been proven. In this paper, we design a cache performance model to analyze the optimality of caching for multimedia streaming environments and design a theoretically optimal caching algorithm based on interval caching. Then, we show that the algorithm we design is an optimal algorithm that minimizes cache misses of streaming data based on the proposed model.

**Key Words :** Multimedia streaming, cache performance model, interval caching, optimality, caching algorithm

## Ⅰ. 서 론

캐쉬 기술은 웹 서버[1], 임베디드 시스템[2], 모바일 엣지 컴퓨팅[3], 스마트 디바이스[4] 등 다양한 컴퓨팅 환경에서 널리 연구되어 왔다. 이는 느린 스토리지 접근 또는 원격 시스템을 통한 데이터 접근 시 발생하는 시간 지연을 해소하기 위해 필수적이다[5, 6, 7]. 한편, 멀티미디어 데이터의 캐쉬은 스트리밍 시스템의 성능에 크게 영향을 미치기 때문에 수십년간 캐쉬 교체 알고리즘에 관한 연구가 진행되었으며, 대부분의 연구가 인터벌 캐쉬의 개념에 기반하고 있다[8, 9, 10]. 인터벌 캐쉬은 스트리밍 데이터의 용량이 매우 크고 순차적으로 접근이 이루어진다

*정회원, 이화여자대학교 컴퓨터공학과
**정회원, 이화여자대학교 임베디드소프트웨어연구센터
접수일자 2023년 9월 18일, 수정완료 2023년 9월 30일
게재확정일자 2023년 10월 6일

Received: 18 September, 2023 / Revised: 30 September, 2023 / Accepted: 6 October, 2023
*Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

는 특성을 활용하는 캐싱 기법이다. 인터벌 캐싱이 실험적으로 우수한 성능을 나타낸다는 점은 입증되었지만[8], 그 유효성을 이론적으로 검증한 연구는 아직까지 이루어지지 않고 있다. 본 논문은 인터벌 캐싱의 효과를 캐쉬 미스율 측면에서 이론적으로 증명하고자 한다. 이를 위해 본 논문에서는 멀티미디어 스티리밍 시스템을 위한 캐쉬 성능평가 모델을 설계하고 이 모델에 근거해서 인터벌 캐싱을 새롭게 정의하고 그 최적성을 증명한다.

본 논문의 이후 구성은 다음과 같다. II장에서는 인터벌 캐싱을 위한 멀티미디어 스트리밍 시스템의 캐쉬 성능평가 모델을 기술한다. III장에서는 인터벌 캐싱의 최적성을 보이기 위해 캐쉬 성능평가 모델에 입각한 새로운 인터벌 캐싱 알고리즘을 정의한다. IV장에서는 III장에서 기술한 인터벌 캐싱 알고리즘이 스트리밍 환경에서 캐쉬 미스를 최소화하는 알고리즘임을 증명한다. 끝으로 V장에서는 본 논문의 결론을 제시한다.

## II. 캐쉬 성능평가 모델

본 장에서는 스트리밍 시스템에서 캐쉬 교체 알고리즘의 성능을 평가하기 위해 캐쉬 성능평가 모델을 설계한다. 본 캐쉬 성능평가 모델에서는 각 캐쉬 블록들이 고정 주기 라운드에 기반해서 주기적으로 스트리밍되며, 캐쉬 블록들은 동일 크기의 파일 블록으로부터 매 라운드 적재되는 것으로 가정한다. 동일 파일에 대한 두 인접 스트림은 인터벌을 형성하며, 인터벌은 두 스트림에 의해 참조되는 파일 블록 간의 거리, 즉 오프셋 차이를 뜻한다. 인터벌을 형성하는 두 스트림은 파일 블록의 접근 위치에 따라 선행 스트림과 후속 스트림으로 나뉘며, 후속 스트림의 인터벌은 이미 선행 스트림에 의해 접근된 파일 블록들로 구성되므로 이를 캐쉬 블록을 통해 최대한 서비스하는 것이 성능 개선의 목표가 된다.

예를 들어, 그림 1과 같이 두 연속적인 스트림 $S_1$과 $S_2$가 인터벌 $I_{21}$을 형성한다고 하자. 이 예에서 스트림 $S_2$는 인터벌 $I_{21}$의 선행 스트림이며, 동시에 인터벌 $I_{32}$의 후속 스트림이다. 그림에서 보는 것처럼 $S_2$는 다음 라운드에서 캐쉬 미스를 발생시키는 반면 $S_1$은 인터벌 $I_{21}$에 속한 캐쉬 블록들로부터 서비스된다. 이와 같은 인터벌 개념은 멀티미디어 시스템에서의 캐싱 알고리즘 설계를 위한 기본적인 아이디어로 활용된다[11, 12, 13]. 즉, 캐쉬 공간이 한정돼 있으므로 인터벌 캐싱은 각 인터벌들을 캐쉬 공간 요구량 순으로 정렬하고 가장 짧은 인터벌부터 캐싱의 대상으로 선정한다[8].

본 논문에서는 인터벌의 개념을 캐쉬 성능평가를 위한 이론적인 모델로 활용하며, 각 라운드마다 발생하는 캐쉬 미스의 합을 성능 평가의 척도로 사용한다. 좀 더 구체적으로는 각 스트림의 확률적인(stochastic) 미스 횟수를 인터벌의 개념을 통해 계산한다. 캐쉬 블록들은 대응하는 파일 블록들이 어떤 인터벌에 포함되는지에 따라 그룹핑할 수 있다. 각 인터벌은 캐쉬 블록들을 담는 컨테이너로 생각할 수 있으며, 캐쉬 미스 횟수는 각 인터벌이 보유할 수 있는 최대 캐쉬 블록 수 대비 현재 할당된 캐쉬 블록의 수를 통해 예측할 수 있다. 본 논문에서는 전자를 인터벌 크기라고 정의하고 후자를 인터벌 할당치라고 정의한다. 이때, 인터벌 크기는 각 라운드와 무관하게 정의되는 개념인 반면 인터벌 할당치는 라운드마다 가변적인 개념이다.

시스템 내에 여러 인터벌이 공존할 수 있으며, 이들을 인터벌 크기 순으로 정렬하는 것이 가능하다. 정렬된 인터벌 리스트를 $(I_1\ I_2\ \dots\ I_n)$라고 할 때, 인터벌 크기 리스트 $\Psi = (\varphi_1\ \varphi_2\ \dots\ \varphi_n)$를 구성할 수 있으며, 이때 $\varphi_x$는 인터벌 $I_x$ $(1 \le x \le n)$의 인터벌 크기를 나타낸다. 또한, 라운드 $t$에서 각 인터벌의 캐쉬 할당 상황을 나타내는 인터벌 할당 리스트를 $\Phi = (\phi_1^t\ \phi_2^t\ \dots\ \phi_n^t)$로 정의할 수 있다. 이와 같은 두 리스트로부터 각 라운드마다 발생하는 캐쉬 미스 카운트를 확률적으로 유도할 수 있다.

라운드 $t$에서 인터벌 크기 리스트 $\Psi$와 인터벌 할당 리스트 $\Phi$가 주어질 때, 캐싱 알고리즘 $\mathbb{A}$에 의해 라운드 $t+j$에서 $t+k$까지 발생하는 캐쉬 미스의 수를 $C_j^k(\mathbb{A}, \Psi, \Phi)$라고 하자. 그러면 캐쉬 미스의 수는 각 라운드별로 인터벌 크기 리스트와 인터벌 할당 리스트에 의해 예측할 수 있다.

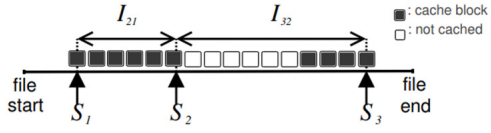


**그림 1. 연속적인 스트림에 의한 인터벌 형성의 예**
**Fig. 1. An example of forming intervals based on consecutive streams.**

## III. 성능평가 모델을 위한 캐싱 정책

본 논문의 캐쉬 성능평가 모델을 위해 이론적으로 최적의 캐싱 알고리즘을 인터벌 캐싱의 개념에 기반해서

새롭게 정의하였으며 이를 RIC(Revised Interval Caching)으로 명명하였다. RIC은 실제 시스템에서의 동작 방식이 기존 인터벌 캐싱과 유사하게 정의되며, 캐쉬 성능평가 모델에서 최적성(optimality)을 증명하기 위해서 이론적으로 고안된 개념이다. RIC의 동작 방식을 이론적으로 보여주기 위해 인터벌 크기 리스트가 ($\varphi_1$ $\varphi_2$ ... $\varphi_n$)로 주어졌을 때 RIC의 라운드 $k$ 에서의 인터벌 할당 리스트가 ( $\phi_1^k$ $\phi_2^k$ ... $\phi_n^k$ )이라고 하자. 그러면, RIC은 다음 조건을 만족하도록 동작한다.

$$\Delta_m^k = \sum_{x=1}^{m-1} \phi_x^k - \sum_{x=m+1}^{n} \left(1 - \left\lceil \frac{\phi_x^k}{\psi_x} \right\rceil\right), \quad 1 \le m \le n. \tag{1}$$

$$\phi_m^{k+1} = \begin{cases} \phi_m^k + 1 & \Delta_m^k > 0 \text{ and } \phi_m^k < \psi_m \\ \phi_m^k + \Delta_m^k & \Delta_m^k < 0 \text{ and } \phi_m^k + \Delta_m^k \ge 0 \\ 0 & \Delta_m^k < 0 \text{ and } \phi_m^k + \Delta_m^k < 0 \\ \phi_m^k & \text{otherwise} \end{cases} \tag{2}$$

위 식에서 $\triangle^k{}_m$는 인터벌 크기가 $\varphi_m$보다 큰 인터벌에게 할당된 캐쉬 블록의 수와 인터벌 크기가 $\varphi_m$ 보다 작은 인터벌에게 할당되지 않은 캐쉬 블록의 수 사이의 차이를 나타낸다. RIC은 인터벌 크기가 작은 인터벌에게 우선적으로 캐쉬 블록을 할당하기 때문에 인터벌 크기가 더 작은 인터벌이 최대로 할당되지 않은 상태에서 인터벌 크기가 더 큰 인터벌에 할당된 캐쉬 블록이 존재할 경우 이를 인터벌 크기가 더 작은 인터벌로 이양한다. 따라서, 순서화된 인터벌 리스트가 주어지면 RIC은 캐쉬 블록들을 인터벌 크기가 작은 인터벌부터 할당하며, 시간이 흐름에 따라 새로운 인터벌이 생성되거나 기존의 인터벌이 사라질 경우 각 인터벌에 할당된 캐쉬 블록들은 이에 근거해서 새롭게 조절된다.

매 라운드마다 RIC은 인터벌 크기가 작은 인터벌에 우선적으로 캐쉬 블록을 할당하므로 빈 캐쉬 블록이 필요할 경우 인터벌 크기가 가장 큰 인터벌의 캐쉬 블록부터 우선적으로 회수한다. 따라서, 일부 인터벌이 특정 라운드에서는 캐쉬 블록을 최대로 할당받았다가 다른 라운드에서는 그렇지 않을 수 있으며, 이는 매 라운드마다 인터벌 크기가 작은 인터벌들이 얼마나 존재하는지에 좌우된다.

서비스 라운드가 진행됨에 따라 RIC이 관리하는 인터벌들은 캐쉬 블록을 모두 할당받은 작은 크기의 인터벌 그룹과 캐쉬 블록을 할당받지 못한 큰 크기의 인터벌 그룹으로 나뉘게 된다. 이때, 캐쉬 블록을 전혀 할당받지 못한 인터벌 크기 $I_\alpha$에 대해 $\alpha$가 순서 리스트 상의 할당 인덱스라고 할 때 RIC은 다음 식을 만족한다.

$$\sum_{x=\alpha+1}^{n} \psi_x < N \le \sum_{x=\alpha}^{n} \psi_x \tag{3}$$

이때, $N$은 캐쉬 블록의 수를 나타내며 이와 같은 상황에서의 할당 리스트를 본 논문에서는 SIFA (Shorter Interval First Allocated) 리스트라고 부른다.

## IV. 최적성의 증명

본 장에서는 RIC 알고리즘이 멀티미디어 스트리밍 환경에서 캐쉬 미스를 최소화하는 알고리즘임을 보인다.

**정리 1.** 인터벌 크기 리스트 $\Psi$에 대한 SIFA 리스트가 $\Phi^s$라고 하자. 그러면 임의의 할당 리스트 $\Phi$에 대해,

$$C_0^1(\mathbb{A}, \Psi, \Phi) - C_0^1(\mathbb{RIC}, \Psi, \Phi^s) \ge 0 \tag{4}$$

가 성립한다.

증명. 정렬된 인터벌 리스트 ($I_1$ $I_2$ ... $I_n$)에 대해 인터벌 크기 리스트와 SIFA 리스트, 인터벌 할당리스트가 각각 $\Psi$ = ($\varphi_1$ $\varphi_2$ ... $\varphi_n$), $\Phi^s$ = ( $\phi_1^s$ $\phi_2^s$ ... $\phi_n^s$), $\Phi$ = ($\varphi_1$ $\varphi_2$ ... $\varphi_n$)이라고 하자. 각 인터벌 $I_x$에 대한 캐쉬 미스의 기대값이 $1-\varphi_x/\varphi_x$이므로 다음이 성립한다.

$$\begin{aligned} C_0^1(\mathbb{A}, \Psi, \Phi) - C_0^1(\mathbb{RIC}, \Psi, \Phi^s) &= \sum_{x=1}^{n} \left(1 - \frac{\phi_x}{\psi_x}\right) - \sum_{x=1}^{n} \left(1 - \frac{\phi_x^s}{\psi_x}\right) \\ &= \sum_{x=1}^{n} \left(\frac{\phi_x^s - \phi_x}{\psi_x}\right) \end{aligned} \tag{5}$$

식 (5)에서 $\phi_x^{diff} = \phi_x^s - \phi_x$라 할 때, 캐쉬 할당 인덱스 $\alpha$에 대해 아래의 결과를 얻을 수 있다.

$$\sum_{x=1}^{n} \frac{\phi_x^{diff}}{\psi_x} = \sum_{x=1}^{\alpha-1} \frac{\phi_x^{diff}}{\psi_x} + \frac{\phi_\alpha^{diff}}{\psi_\alpha} + \sum_{x=\alpha+1}^{n} \frac{\phi_x^{diff}}{\psi_x} \tag{6}$$

식 (6)에서 $\phi_x^{diff}$는 $x<\alpha$인 경우 0 이하의 값을 가지며 $x>\alpha$인 경우 0 이상의 값을 가진다. 따라서, 아래 식을 도출할 수 있다.

$$\sum_{x=1}^{\alpha-1} \frac{\phi_x^{diff}}{\psi_x} \ge \frac{\sum_{x=1}^{\alpha-1} \phi_x^{diff}}{\psi_\alpha}, \quad \sum_{x=\alpha+1}^{n} \frac{\phi_x^{diff}}{\psi_x} \ge \frac{\sum_{x=\alpha+1}^{n} \phi_x^{diff}}{\psi_{\alpha+1}} \tag{7}$$

또한 이를 통해 식 (8)을 유도할 수 있다.

$$\sum_{x=1}^{n}\frac{\phi_x^{diff}}{\psi_x} \geq \frac{\sum_{x=1}^{\alpha-1}\phi_x^{diff}}{\psi_\alpha} + \frac{\phi_\alpha^{diff}}{\psi_\alpha} + \frac{\sum_{x=\alpha+1}^{n}\phi_x^{diff}}{\psi_{\alpha+1}} \quad (8)$$

식 (8)에

$$\sum_{x=1}^{n}\phi_x^{diff} = 0 \quad (9)$$

를 적용하면 아래 식을 얻을 수 있다.

$$\begin{aligned}\sum_{x=1}^{n}\frac{\phi_x^{diff}}{\psi_x} &\geq \frac{\sum_{x=1}^{\alpha-1}\phi_x^{diff}}{\psi_\alpha} + \frac{\phi_\alpha^{diff}}{\psi_\alpha} + \frac{\sum_{x=\alpha+1}^{n}\phi_x^{diff}}{\psi_{\alpha+1}} \\ &= \frac{-\sum_{x=\alpha}^{n}\phi_x^{diff}}{\psi_\alpha} + \frac{\phi_\alpha^{diff}}{\psi_\alpha} + \frac{\sum_{x=\alpha+1}^{n}\phi_x^{diff}}{\psi_{\alpha+1}} \\ &= \frac{-\sum_{x=\alpha+1}^{n}\phi_x^{diff}}{\psi_\alpha} + \frac{\sum_{x=\alpha+1}^{n}\phi_x^{diff}}{\psi_{\alpha+1}} \\ &= \frac{\sum_{x=\alpha+1}^{n}\phi_x^{diff}\cdot(\psi_\alpha - \psi_{\alpha+1})}{\psi_\alpha\cdot\psi_{\alpha+1}} \geq 0\end{aligned} \quad (10)$$

**정리 2.** 임의의 캐싱 알고리즘 $\mathbb{A}$에 대해 다음 식을 만족하는 $c$ ($c \leq j$)가 존재한다.

$$C_j^k(\mathbb{A},\Psi,\Phi) - C_j^k(\mathbb{RIC},\Psi,\Phi) \geq 0 \quad (11)$$

증명. $\Phi^{\mathbb{A}}$과 $\Phi^{\mathbb{RIC}}$가 각각 알고리즘 $\mathbb{A}$와 $\mathbb{RIC}$이 캐쉬를 $j$ 라운드 동안 운영한 이후의 할당 리스트라 하고, $\Phi_x^{\mathbb{A}}$과 $\Phi_x^{\mathbb{RIC}}$를 $j$ 라운드부터 시작해서 $x$ 라운드까지 서비스된 시점의 할당 리스트라고 하자. 그러면 $\Phi^{\mathbb{RIC}}$가 SIFA 리스트가 되도록 $c$를 잡을 수 있다. 따라서 모든 $\Phi_x^{\mathbb{RIC}}$가 또한 SIFA 리스트이다. 따라서, 아래 식이 도출된다.

$$\begin{aligned}&C_j^k(\mathbb{A},\Psi,\Phi) - C_j^k(\mathbb{RIC},\Psi,\Phi) \\ &= C_0^{k-j}(\mathbb{A},\Psi,\Phi^{\mathbb{A}}) - C_0^{k-j}(\mathbb{RIC},\Psi,\Phi^{\mathbb{RIC}}) \\ &= \sum_{x=1}^{k-j} C_0^1(\mathbb{A},\Psi,\Phi_x^{\mathbb{A}}) - \sum_{x=1}^{k-j} C_0^1(\mathbb{RIC},\Psi,\Phi_x^{\mathbb{RIC}}) \\ &= \sum_{x=1}^{k-j}\left(C_0^1(\mathbb{A},\Psi,\Phi_x^{\mathbb{A}}) - C_0^1(\mathbb{RIC},\Psi,\Phi_x^{\mathbb{RIC}})\right)\end{aligned} \quad (12)$$

식 (12)의 모든 항이 정리 1에 의해 0 이하의 값을 가지므로 $C_j^k(\mathbb{A},\Psi,\Phi) - C_j^k(\mathbb{RIC},\Psi,\Phi)$ 역시 0 이하의 값을 가진다.

## V. 결 론

멀티미디어 스트리밍 서버 환경을 위한 많은 캐싱 알고리즘이 제안되었으나, 아직까지 LRU (least recently used) 알고리즘이 여전히 실제 시스템에서 많이 사용되고 있다. 이는 대부분의 개발자가 파일 시스템의 버퍼 캐싱 기법으로 멀티미디어 데이터를 캐싱하는 것이 여전히 효과적이라고 생각하기 때문이다. 한편, 스트리밍 데이터에 대해서는 인터벌 캐싱에 기반한 캐쉬 교체 알고리즘이 캐싱의 효과를 높일 수 있음이 실험적으로 입증된 바 있다. 본 논문에서는 이러한 인터벌 캐싱이 이론적으로 우수한 성능을 가짐을 증명하기 위해 캐쉬 성능평가 모델을 설계하고, 이론적인 인터벌 캐싱 알고리즘인 RIC이 스트리밍 환경을 위한 성능 평가 모델에서 캐쉬 미스를 최소화하는 최적의 교체 알고리즘임을 보였다.

## References


[1] H. Bahn, H. Lee, S.H. Noh, S.L. Min, K. Koh, "Replica-aware caching for web proxies," Computer Communications, vol. 25, no. 3, pp. 183-188, 2002. DOI: https://doi.org/10.1016/S0140-3664(01)00365-6

[2] T. Kim and H Bahn, "Implementation of the storage manager for an IPTV set-top box," IEEE Transactions on Consumer Electronics, vol. 54, no. 4, pp. 1770-1775, 2008. DOI: https://doi.org/10.1109/TCE.2008.4711233

[3] S. Qiu, Q. Fan, X. Li, X. Zhang, G. Min, and Y. Lyu, "OA-Cache: Oracle Approximation based Cache Replacement at the Network Edge," IEEE Transactions on Network and Service Management, Early access, 2023. DOI: https://doi.org/10.1109/TNSM.2023.3239664.

[4] J. Zhang, M. Lin, Y. Pan, and Z. Xu, "CRFTL: Cache Reallocation-Based Page-Level Flash Translation Layer for Smartphones," IEEE Transactions on Consumer Electronics, vol. 69, no. 3, pp. 671-679, 2023. DOI: https://doi.org/10.1109/TCE.2023.3264217

[5] B. Kim, Y. Woo, and I. Kim, "Aggressive Cooperative Caching Scheme in Mobile Ad Hoc Networks," Journal of KIIT, vol. 18, no. 1, pp. 41-48, 2020. DOI: https://doi.org/10.14801/jkiit.2020.18.1.41

[6] P. Kwon and S. Ahn, "Dynamic Bandwidth Distribution Method for High Performance Non-volatile Memory in Cloud Computing Environment," The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), vol. 20, no. 3, pp. 97-103, 2020.
DOI: https://doi.org/10.7236/JIIBC.2020.20.3.97

[7] S. Park, "RFJ: A reliable and fast journaling mechanism," Journal of the Korea Academia-Industrial cooperation Society(JKAIS), vol. 20, no. 7, pp. 45-51, 2019.
DOI: https://doi.org/10.5762/KAIS.2019.20.7.45.

[8] A. Dan and D. Sitaram, "Buffer Management Policy for an On-Demand Video Server," IBM Research Report, RC 19347, Yorktown Heights, NY, 1993.

[9] A. Dan, Y. Heights, and D. Sitram, "Generalized interval caching policy for mixed interactive and long video workloads," In Proc. SPIE Conf. Multimedia Computing and Networking, 1996.
DOI: https://doi.org/10.1117/12.235887

[10] T. Kim, H. Bahn, and K. Koh, "Popularity-aware interval caching for multimedia streaming servers," Electronics Letters, vol. 39, no. 21, pp. 1555-1557, 2003.
DOI: https://doi.org/10.1049/el:20030965

[11] O. Kwon, H. Bahn, and K. Koh, "Popularity and prefix aware interval caching for multimedia streaming servers," in Proc. 8th IEEE International Conference on Computer and Information Technology, pp. 555-560, 2008.
DOI: https://doi.org/10.1109/CIT.2008.4594735

[12] L. Dong and B. Veeravalli, "Design and analysis of a variable bit rate caching algorithm for continuous media data," Multimedia Tools and Applications, vol. 38, no. 1, pp. 91-117, 2008.
DOI: https://doi.org/10.1007/s11042-007-0151-6

[13] L. Wujuan, L.S. Yong and Y.K. Leong, "A client-assisted interval caching strategy for video-on-demand systems," Computer Communications, vol. 29, no. 18, pp. 3780-3788, 2006.
DOI: https://doi.org/10.1016/j.comcom.2006.06.014

## 저 자 소 개

**반 효 경(정회원)**

- 1997년 2월 : 서울대학교 계산통계학과 학사
- 1999년 2월 : 서울대학교 전산과학과 석사
- 2002년 2월 : 서울대학교 컴퓨터공학부 박사.
- 2002년 9월 ~ : 이화여자대학교 컴퓨터공학과 교수.
- 주관심분야 : 운영체제, 스토리지시스템, 임베디드시스템

**조 경 운(정회원)**

- 1995년 2월 : 서울대학교 계산통계학과 학사
- 1997년 2월 : 서울대학교 전산과학과 석사
- 2012년 2월 : 서울대학교 컴퓨터공학부 박사
- 2000년 ~ 2016년 : ㈜클루닉스 연구소장
- 2016년 4월 ~ : 이화여자대학교 임베디드소프트웨어연구센터 수석연구원/연구교수

※ This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MOE) (No. RS-2023-00247727) and by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub).