Ko et al. EURASIP Journal on Wireless Communications and Networking (2018) 2018:190
https://doi.org/10.1186/s13638-018-1204-4

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Energy-efficient proactive edge caching with sleep scheduling for green networks

Haneul Ko[1,2], Jaewook Lee[3] and Sangheon Pack[3*]

Abstract

Proactive content caching in small cells (SCs) (i.e., proactive edge caching) can significantly reduce energy consumption of networks. However, the sleep mode of SCs can make the cached contents in SCs unavailable. Therefore, a joint optimization of edge caching strategy and sleep scheduling should be conducted to maximize the effectiveness of edge caching. In this paper, we propose an energy-efficient proactive edge caching with sleep scheduling (E3CS) where the controller jointly adjusts the caching strategy and the sleep scheduling of SCs by considering the content popularity dynamics. To optimize the performance of E3CS (i.e., minimize the overall energy consumption), we formulate a Markov decision process (MDP) and the joint optimal policy on the caching strategy and sleep scheduling is obtained by a value iteration algorithm. Evaluation results demonstrate that E3CS with the optimal policy outperforms the comparison schemes in terms of the overall energy consumption.

Keywords: Content caching, Edge caching, Sleep scheduling, Energy, Markov decision process (MDP), Green networks

1 Introduction

Recent report [1] predicts that mobile data traffic will grow at a compound annual growth rate of 47% from 2016 to 2021. In addition, it shows that most Internet traffic is governed by content retrieval applications such as content sharing and video on demand (VOD) services. In these applications, users retrieve contents from a remote content server, which causes a significant amount of traffic and energy consumption. To alleviate this problem, the content caching where contents are stored in intermediate nodes of networks has received a high attention by lots of network operators. By amplifying the caching efficiency, contents can be proactively stored on the network edge (e.g., small cells (SCs)) [2–5].[1]

Specifically, the energy consumption for content transmission over the backhaul link can be significantly reduced by means of proactive edge caching in SCs. Meanwhile, SCs usually use the sleep mode to reduce the maintenance energy needed to be active. However, it can make the cached contents in SCs (in the sleep mode) unavailable. In this case, the contents should be delivered through the backhaul link from the remote content server even though the contents are stored in SCs, which increases the energy consumption. Therefore, the caching strategy and sleep scheduling should be jointly adjusted to reduce the overall energy consumption. However, only few works consider simultaneously both the caching strategy and sleep scheduling [6, 7].

The popularity of contents changes dynamically with temporal patterns [8]. Specifically, contents on microblogging platform such as Facebook and Twitter are very volatile, and they become popular and fade away in a matter of minutes. Moreover, their popularity can exhibit a variety of patterns [9]. Obviously, when the popular contents are cached, the utility of the cached contents (e.g., hit ratio) increases. Meanwhile, the content popularity dynamics have strong correlation between short-term popularity and long-term popularity, and thus, it can be well-estimated by several methods [10–15]. However, most of existing cache systems are not designed by considering the content popularity dynamics, which is reasonable only when the cache churn time is small compared with the popularity dynamics.

2 Methods

In this paper, we propose an energy-efficient proactive edge caching with sleep scheduling (E3CS) where the

*Correspondence: shpack@korea.ac.kr
[3]School of Electrical Engineering, Korea University, Anam-ro 145, Seoul, Korea
Full list of author information is available at the end of the article

controller jointly adjusts the caching strategy and the sleep scheduling of SCs by considering the content popularity dynamics. Specifically, the controller does not command the sleep mode to SCs having high utility of the cached contents (e.g., popular contents) and caches proactively contents estimated to become popular in SCs to reduce the energy consumption for the content delivery. Moreover, the controller commands the sleep mode to SCs with low utility of the cached contents (e.g., when there is no user in the coverage of SC). Then, the energy consumption to maintain the active mode of SCs can be diminished. To optimize the performance of E3CS (i.e., minimize the overall energy consumption), we formulate a Markov decision process (MDP) and the joint optimal policy on the caching strategy and sleep scheduling is obtained by a value iteration algorithm. Evaluation results demonstrate that E3CS with the optimal policy can reduce the overall energy consumption by up to 25%.

The contribution of this paper can be summarized as follows: (1) to the best of our knowledge, this is the first work to simultaneously optimize the caching strategy and sleep scheduling with the consideration of the content popularity dynamics and (2) we present and analyze evaluation results under various environments, which provide guidelines for designing the edge caching system with high energy efficiency.

The remainder of this paper is organized as follows. Related works are summarized in Section 3 and E3CS is described in Section 4. After that, the MDP model is developed in Section 5 and the evaluation results are given in Section 6, and followed by the concluding remarks in Section 7.

3 Related works

To improve the energy efficiency of networks by means of the content caching, there are several works in the literature [6, 7, 10–26].

To maximize the effectiveness of caching, sociality between users is exploited in [16, 17]. Nikolaou et al. [16] proposed two cache placement strategies that take advantage of known relationships between clients (e.g., social links) and the workload on the service. In a similar context, Wang et al. [17] analyzed the impact of social relationships on the performance of edge caching by means of a Markov chain. Meanwhile, Kakar et al. [18] introduced the concept of delivery time per bit and investigated cache-aided heterogeneous wireless networks. Since the performance of the caching system can be improved when the coded contents are cached, some works focused on the coded contents and introduced appropriate caching schemes for these contents [19–21]. Maddah-Ali and Niesen [19] proposed a coded caching scheme to improve substantially the efficiency of content transmission over un-coded caching. Fadlallah et al. [20] provided an overview of some caching-aided coded multicast techniques and then discussed the potential of caching-aided coded multicast for improving bandwidth efficiency. Poularakis et al. [21] developed caching algorithms for layered encoding (i.e., scalable video coding (SVC)) to reduce the average video delivery time. In [22, 23], it is assumed that the content popularity distribution is fixed, which follows a specific distribution (i.e., Zipf distribution). Specifically, Zhou et al. [22] designed a cache strategy called caching as a cluster where SCs can exchange contents with each other within the cluster of SCs, and a probabilistic solution was derived based on the content popularity distribution to reduce the average content delivery latency. Meanwhile, Krishnan et al. [23] studied the effect of retransmissions on the cache strategy for both static and mobile user scenarios with the fixed content popularity distribution and demonstrated that the cache strategy is very sensitive to the number of retransmissions. Since the content popularity is fluctuated, Muller et al. [24] suggested an algorithm which learns context-specific content popularity by regularly observing context information of users and updating the cached contents. However, these works did not take into account the effect of SCs' sleep mode on the performance of the content caching.

Some works analyzed the energy efficiency of edge caching-enabled networks [25, 26]. Liu and Yang [25] derived a closed form of the energy efficiency and identified key impacting factors to the energy efficiency. Similarly, Perabathini et al. [26] analyzed the energy efficiency and area power consumption based on stochastic geometry and provided the optimal transmission power to maximize the energy efficiency.

Since the popular contents should be cached to improve the performance of the caching system, it is important to estimate the content popularity dynamics. Therefore, some works focused on the content popularity dynamics estimation [10–15]. Yang and Leskovec [10] studied on temporal patterns of contents and how the content popularity grows and fades over time. Similarly, Zhou et al. [11] analyzed how the content popularity changes with time, for different types of contents, and then apply the results to design caching strategies. Meanwhile, some works developed a prediction model based on empirical data [12–14]. Szabo and Huberman [12] presented a model that predicts the long-term content popularity based on early measurements of user access. Li et al. [13] proposed a model that can capture the popularity dynamics based on popularity characteristics (e.g., early popularity evolution pattern and future popularity burst possibility). Similarly, Traverso et al. [14] devised a simple model to capture the dynamics of content popularity. Since most existing popularity prediction models are based on the big data analytics, their accuracy is highly

dependent on the size of data sets. To mitigate this problem, He et al. [15] predicted the viewing probability of contents from the perspective of individuals by means of a discrete-time Markov chain.

Few works considered simultaneously both the caching strategy and the sleep scheduling [6, 7]. Li et al. [6] constructed the framework of edge caching with considering the interaction between caching and sleeping. Specifically, the content caching problem was formulated to maximize the average hit rate with an energy consumption constraint. Meanwhile, Chiang et al. [7] investigated a multi-cell cooperation-based approach where a cooperative transmission and sleep mode operation are jointly performed to tackle inter-cell interference while reducing energy dissipation. However, since these works did not consider the content popularity dynamics, the optimal performance of edge caching system cannot be achieved.

4 Energy-efficient proactive edge caching with sleep scheduling (E3CS)

Figure 1 shows the system model in this paper. We consider a heterogenous networks where a macro cell (MC) overlays with several SCs which are connected to MC through an interface (e.g., X2 interface in LTE/LTE-A networks).

Two types of energy consumption are considered in our system model: (1) the energy consumption to deliver the contents and (2) the energy consumption to maintain the active mode of SCs. The energy consumption to deliver the contents can be reduced by the edge caching. When a cached content is requested, SC can provide it and thus the energy consumption owing to backhaul link transmissions can be reduced.[2] To improve this effect, the popular contents should be previously cached before when these are requested. That is, contents should be proactively cached with the consideration of the content popularity dynamics (i.e., contents should be cached before those become popular).

Meanwhile, we assume that SCs in the active mode consume E_A energy during the unit time τ. To reduce this energy consumption, SCs can use the sleep mode. Since MC generally provides control messages to users, we assume that MC in our system model does not use the sleep mode. In the sleep mode, since most functions of SCs are turned off, the energy consumption in this mode can be negligible. However, the sleep mode makes the cached contents unavailable. Therefore, the contents should be delivered from the remote content server, which increases the energy consumption to deliver the contents. Especially when SC users in the sleep mode request excessively, huge energy can be consumed. Therefore, SCs use the sleep mode only when the utility of the cached contents is low (e.g., there is no user in the coverage of SC).

In E3CS, the controller at MC side jointly decides the caching strategy and conducts the sleep scheduling of SCs to minimize the overall energy consumption. For this, the controller maintains some information such as the content popularity dynamics and the number of users (i.e., mobility profiles of users) in each SC. Note that the content request frequency is dependent on the number of users in SCs. By using these information, the controller makes the policy table on the caching strategy and sleep scheduling and distributes the table to SCs. Basically, we assume that the library of candidate contents to be cached is not changed in a short-term scenario,[3] and therefore, we can obtain the optimal policy in an offline manner. Since newly generated contents are generally not popular

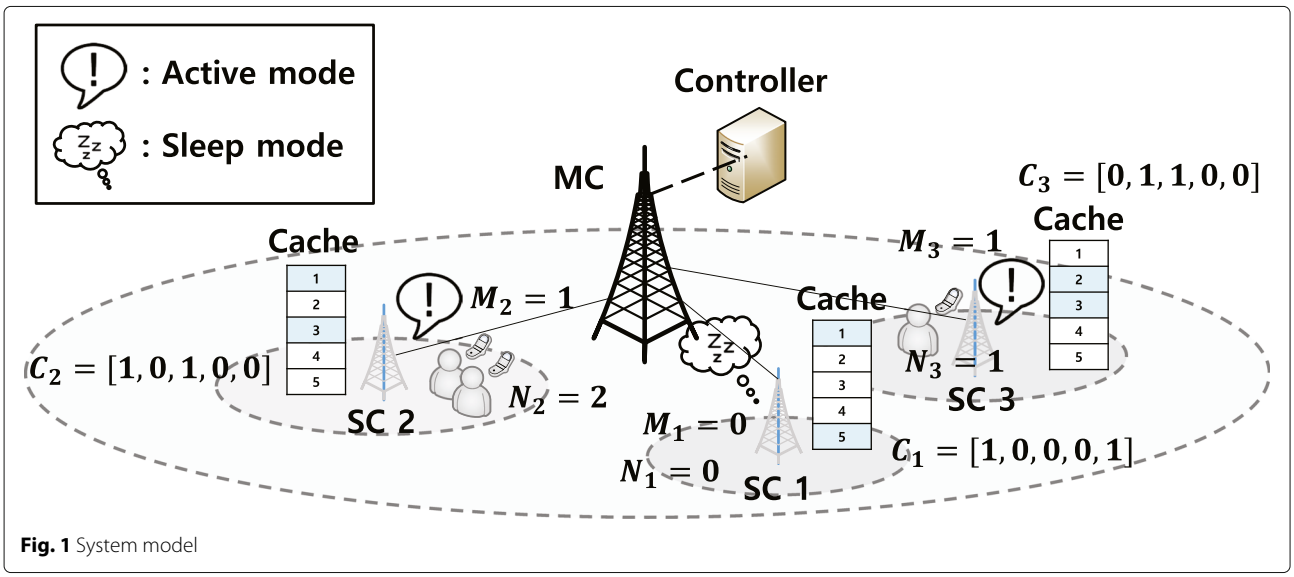

Fig. 1 System model

at the first time, these contents do not need to be cached in SCs. Therefore, we believe that our caching scenario is reasonable. Meanwhile, when some parameters (e.g., content popularity) are changed or new contents become popular, a new policy table should be reconstructed by the controller to achieve better performance. Since the complexity of making a policy table is not negligible, too frequent reconstructions can be huge overhead to the controller. Therefore, an appropriate frequency should be determined to balance the tradeoff between the performance improvement and the overhead, which is one of our future works.[4] After receiving the policy table, SCs decide which contents are cached in their storages and whether to sleep or not by following the distributed policy table. To obtain the optimal policy table, we formulate MDP which will be elaborated in Section 5. Since MC has higher computational power than SCs and the policy table is made by the controller at MC side, the MDP model can be applied to SCs without any high computation overhead in SCs.

5 Markov decision process (MDP)

To construct the optimal policy table on the caching strategy and sleep scheduling, we formulate an MDP model in this section. The MDP model represents a mathematical framework to model decision making in situations in which outcomes are partially random and partially under the control of the decision maker [27]. Therefore, the MDP model appears suitable for simultaneously adjusting the cache strategy and sleep scheduling under the content popularity dynamics. The MDP model is constructed with five elements: (1) decision epoch, (2) action, (3) state, (4) transition probability, and (5) cost functions [28, 29]. Subsequently, we introduce the optimality equation and a value iteration algorithm to solve the equation. Important notations for the MDP model are summarized in Table 1.

Table 1 Summary of notations

Notation	Description
$\mathbf{S_t}$	State at the decision epoch t
$\mathbf{A_t}$	Action chosen at the decision epoch t
τ	Duration of each decision epoch
$\mathbb{S}$	State space
$\mathbb{P}$	State space for denoting the popularity of contents
$\mathbb{M}$	State space for denoting the mode of SCs
$\mathbb{N}$	State space for denoting the number of users in SCs
$\mathbb{C}_i$	State space for denoting cached contents in SC i
N_S	Number of SCs
N_C	Number of contents
M_U	Maximum number of users in one SC
$\mathbb{A}$	Action space
$\mathbf{A^M}$	Action vector for deciding mode
$\mathbf{A^C}$	Action vector for deciding whether each content is cached at SCs
ξ	Maximum number of contents which can be cached in SCs
E_A	Energy consumption of the active mode during τ
N_R^{ij}	Expected number of requests for content j in SC i
λ	Discount factor

5.1 Decision epoch

Figure 2 shows the timing diagram for the MDP model. A sequence $T = \{1, 2, 3, \ldots\}$ represents the time epochs when successive decisions are made. $\mathbf{S_t}$ and $\mathbf{A_t}$ denote the state and the action chosen at the decision epoch $t \in T$, respectively. τ represents the duration of each decision epoch.

5.2 State space

We define the state space $\mathbb{S}$ as

$$\mathbb{S} = \mathbb{P} \times \mathbb{M} \times \mathbb{N} \times \prod_{i}^{N_S} \mathbb{C}_i \tag{1}$$

where $\mathbb{P}$ and $\mathbb{M}$ denote the state spaces for representing the popularity of contents and the mode of SCs, respectively. In addition, $\mathbb{N}$ means the state space for describing the number of users in SCs. Moreover, $\mathbb{C}_i$ describes the state space for representing cached contents of SC i. Meanwhile, N_S is the number of SCs.

$\mathbb{P}$ can be represented as

$$\mathbb{P} = \prod_{j}^{N_C} P_j \tag{2}$$

where N_C is the number of contents. In addition, P_j denotes the popularity of content j. That is, $P_j = \{0, 1, 2, \ldots, M_P\}$ where M_P is the maximum level of content popularity. Note that the content popularity can be quantized by a certain criterion (e.g., the popularity ranking).

Meanwhile, $\mathbb{M}$ is described by

$$\mathbb{M} = \prod_{i}^{N_S} M_i \tag{3}$$

where M_i denotes the mode of SC i. That is, if SC i is in the sleep mode, $M_i = 0$. Otherwise, $M_i = 1$.

$\mathbb{N}$ can be denoted by

$$\mathbb{N} = \prod_{i}^{N_S} N_i \tag{4}$$

where N_i denotes the number of users in SC i. That is, $N_i = \{0, 1, 2, \ldots, M_U\}$ where M_U is the maximum number of users in one SC.

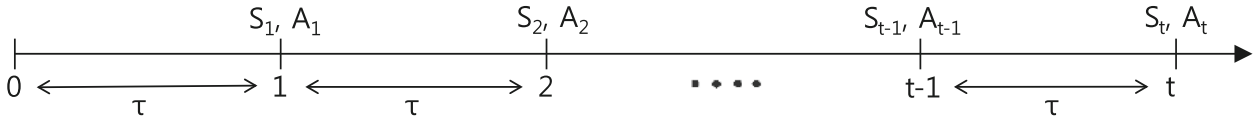

Fig. 2 Timing diagram

On one hand, $\mathbb{C}_i$ can be represented as

$$\mathbb{C}_i = \prod_{j}^{N_C} c_i^j \tag{5}$$

where c_i^j denotes whether the content j is cached in SC i or not. In other words, if the content j is cached in SC i, $c_i^j = 1$. Otherwise, $c_i^j = 0$. The element of $\mathbb{C}_i$ can be represented by the vector $\mathbf{C_i}$. For example, when the total number of contents is 5 and the first and third contents are stored in SC 2, $\mathbf{C_2} = [1, 0, 1, 0, 0]$ as shown in Fig. 1.[5]

5.3 Action space

The action space $\mathbb{A}$ can be described as

$$\mathbb{A} = \{\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_{N_{P,A}}}\} \tag{6}$$

where $N_{P,A}$ means the total number of possible combinations of actions. $\mathbf{A_k}$ represents the kth possible combination of actions. At each decision epoch, the controller determines which SCs are in the sleep mode (i.e., conducts the sleep scheduling). In addition, the controller decides whether each content is cached at each SC (i.e., decides the cache strategy). Therefore, one of the vectors in the action vector set (i.e., $\mathbf{A}$) can be constructed by concatenating two vectors: (1) the action vector for the sleep scheduling, $\mathbf{A^M}$ and (2) the action vector for the cache strategy, $\mathbf{A^C}$. That is, $\mathbf{A} = \mathbf{A^M} \oplus \mathbf{A^C}$ where $\oplus$ means the concatenation operation between two vectors. Moreover, $\mathbf{A^C}$ can be constructed by concatenating N_S vectors where each vector $\mathbf{A_i^C}$ decides whether each content is cached at SC i $\left(\text{i.e., } \mathbf{A^C} = \mathbf{A_1^C} \oplus \mathbf{A_2^C} \oplus \ldots \oplus \mathbf{A_{N_S}^C}\right)$.

The controller commands each SC whether it goes in the sleep mode or not. Therefore, $\mathbf{A^M}$ can be represented by $\left[a_1^M, a_2^M, \ldots, a_{N_S}^M\right]$, where a_i^M denotes the commanded mode (i.e., sleep or active mode) to SC i. That is, if $a_i^M = 0$, the controller commands the sleep mode to SC i. On the contrary, $a_i^M = 1$ means that the controller commands SC i to be active.

Meanwhile, $\mathbf{A_i^C}$ can be denoted as $\left[a_{i,1}^C, a_{i,2}^C, \ldots, a_{i,N_C}^C\right]$ where $a_{i,j}^C$ represents whether SC i tries to cache content j. In other words, $a_{i,j}^C = 1$ means that SC i tries to cache content j (i.e., SC i downloads content j from the content server), whereas $a_{i,j}^C = 0$ means that SC i does not try. Note that the number of contents can be cached is limited due to the constraint of the storage capacity of SCs. Therefore, the possible action vector $\mathbf{A_i^C}$ which can be chosen is restricted. That is, when κ_j denotes the size of content j, only $\mathbf{A_i^C}$ with $\sum_j \kappa_j a_{i,j}^C \leq \xi$ where ξ is the maximum number of contents which can be cached in SCs can be chosen.

5.4 Transition probability

We can think that two type of actions (i.e., action for the sleep scheduling $\mathbf{A^M}$ and action for the cache strategy $\mathbf{A^C}$) are chosen at each decision epoch. Then, cached content state $\mathbf{C}$ and SC mode state $\mathbf{M}$ are dependent on the chosen action $\mathbf{A^C}$ and $\mathbf{A^M}$, respectively. Meanwhile, all states change independently with each other. Therefore, the transition probability from the current state, $\mathbf{S} = [\mathbf{P}, \mathbf{M}, \mathbf{N}, \mathbf{C}]$, to the next state, $\mathbf{S'} = \left[\mathbf{P'}, \mathbf{M'}, \mathbf{N'}, \mathbf{C'}\right]$, can be represented by

$$\begin{aligned} P\left[\mathbf{S'}|\mathbf{S}, \mathbf{A}\right] &= P\left[\mathbf{P'}|\mathbf{P}\right] \times P\left[\mathbf{M'}|\mathbf{M}, \mathbf{A^M}\right] \\ &\quad \times P\left[\mathbf{N'}|\mathbf{N}\right] \times P\left[\mathbf{C'}|\mathbf{C}, \mathbf{A^C}\right]. \end{aligned} \tag{7}$$

The transition probability of each state can be derived as follows. Since the popularity of contents are varied independently, $P\left[\mathbf{P'}|\mathbf{P}\right] = \prod_j^{N_C} P\left[P_j'|P_j\right]$ where P_j and P_j' denote the popularity of content j at the next states, respectively. Meanwhile, $P\left[P_j'|P_j\right]$ can be obtained by various methods such as statistic manner and model-based approach [10–15]. For example, by using shot noise model (SNM) [14], [6] the request rate can be modeled as a function of the time. Since the request rate at a certain time can represent the popularity at that time, $P\left[P_j'|P_j\right]$ can be defined.

Since the modes of SCs are decided independently according to the elements in the chosen action vector $\mathbf{A^M}$, $P\left[\mathbf{M'}|\mathbf{M}\right] = \prod_i^{N_S} P\left[M_i'|M_i, a_i^M\right]$ where M_i and M_i' represent the mode of SC i at the current and next states, respectively. Since M_i' is decided by a_i^M (i.e., SC i changes its mode by following the command of the controller), $P\left[M_i'|M_i, a_i^M = 0\right]$ and $P\left[M_i'|M_i, a_i^M = 1\right]$ can be represented by

$$P\left[M_i'|M_i, a_i^M = 0\right] = \begin{cases} 1, & \text{if } M_i' = 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

and

$$P\left[M_i'|M_i, a_i^M = 1\right] = \begin{cases} 1, & \text{if } M_i' = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Meanwhile, $P\left[\mathbf{N}'|\mathbf{N}\right]$ can be defined by means of a statistical manner or a mathematical assumption (e.g., uniform distribution and point Poisson process [30]). Note that $P\left[\mathbf{N}'|\mathbf{N}\right]$ means that the population distribution of SCs and the mobility of users can be reflected in this probability.

On the other hand, the contents stored in SC i are independently decided by the chosen action vector $\mathbf{A_i^C}$. Therefore, $P\left[\mathbf{C}'|\mathbf{C},\mathbf{A^C}\right] = \prod_{i}^{N_S} P\left[\mathbf{C_i'}|\mathbf{C_i},\mathbf{A_i^C}\right]$ where $\mathbf{C_i}$ and $\mathbf{C_i'}$ represent the cached contents in SC i at the current and next states, respectively. Meanwhile, whether content j is stored in SC i or not is decided according to $a_{i,j}^C$. That is, $P\left[\mathbf{C_i'}|\mathbf{C_i},\mathbf{A_i^C}\right]$ can be calculated by $\prod_{j}^{N_C} P\left[c'^j_i|c_i^j,a_{i,j}^C\right]$ where c_i^j and c'^j_i denote the current and next states for the cached status of content j, respectively. $P\left[c'^j_i|c_i^j,a_{i,j}^C\right]$ is derived as follows. If SC i does not have content j and the controller decides that content j is stored at SC i $\left(\text{i.e., } c_i^j = 0 \text{ and } a_{i,j}^C = 1\right)$, SC i downloads content j by requesting it to the content server. We assume that the download completion time for content j follows an exponential distribution with mean $1/\mu_j$.[7] Then, the probability that content j is stored in SC i at the next state can be calculated by $\mu_j\tau$ [31]. Meanwhile, when SC i does not have content j and the controller decides that content j is not stored at SC i $\left(\text{i.e., } c_i^j = 0 \text{ and } a_{i,j}^C = 0\right)$, any state transition does not occur. Therefore, the corresponding state transition probabilities can be derived as

$$P\left[c'^j_i|c_i^j = 0, a_{i,j}^C = 1\right] = \begin{cases} \mu_j\tau, & \text{if } c'^j_i = 1 \\ 1-\mu_j\tau, & \text{if } c'^j_i = 0. \end{cases} \tag{10}$$

and

$$P\left[c'^j_i|c_i^j = 0, a_{i,j}^C = 0\right] = \begin{cases} 1, & \text{if } c'^j_i = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

When the controller decides that content j is stored at SC i $\left(\text{i.e., } a_{i,j}^C = 1\right)$, if SC i has already content j, $\left(\text{i.e., } c_i^j = 1\right)$, SC i needs not to download content j. Thus, any state transition does not occur. Meanwhile, SC i can delete content j immediately. Therefore, when the controller decides that content j is not stored at SC i $\left(\text{i.e., } a_{i,j}^C = 0\right)$, the next state for content j is 0 $\left(\text{i.e., } c'^j_i = 0\right)$ even though the current state for content j is 1 $\left(\text{i.e., } c_i^j = 1\right)$. Then, the corresponding transition probabilities can be described by

$$P\left[c'^j_i|c_i^j = 1, a_{i,j}^C = 1\right] = \begin{cases} 1, & \text{if } c'^j_i = 1 \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

and

$$P\left[c'^j_i|c_i^j = 1, a_{i,j}^C = 0\right] = \begin{cases} 1, & \text{if } c'^j_i = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

5.5 Cost function

To define the cost function, we consider the overall energy consumption. The overall energy consumption consists of (1) the energy consumption to deliver the contents, (2) the energy consumption to maintain the active mode of SCs, and (3) the energy consumption to change of the cached contents. Since one SC in the active mode consumes E_A energy during τ, the energy consumption to maintain the active mode of SCs can be calculated as $E_A \sum_{i}^{N_S} \delta\left[M_i = 1\right]$, where $\delta[\cdot]$ is a delta function to return 1 if the given condition (e.g., $M_i = 0$) is true. If the condition is not true, a delta function returns 0. Meanwhile, the expected number of requests for content j in SC i at each decision epoch, $N_R^{i,j}$, is proportional to its popularity and the number of users in SC i. That is, $N_R^{i,j}$ can be calculated by $\alpha P_j n_i$ where α is the coefficient parameter reflecting the duration of decision epoch.[8] When the content j is requested, if SC i is in the active mode and has content j in its storage $\left(\text{i.e., } M_i = 1 \text{ and } c_i^j = 1\right)$, SC i can transport content j to users. In this case, a small amount of delivery energy consumption $E_{C,E}$ is needed. On the contrary, when SC i is in the sleep mode or it does not have content j in its storage $\left(\text{i.e., } M_i = 0 \text{ or } c_i^j = 0\right)$, the requested content should be delivered from the content server. Since the energy consumption to deliver the contents is generally proportional to the distance between the source and the destination, a large amount of energy $E_{C,C}$ can be assumed when the content is delivered from the content server. Meanwhile, when SC i tries to change its stored content $\left(\text{i.e., } a_{i,j}^C = 0 \text{ and } a_{i,j}^C = 1 \text{ or } a_{i,j}^C = 1 \text{ and } a_{i,j}^C = 0\right)$, the energy E_S is needed.[9] To sum up, the cost function for the total energy consumption of networks $r(S,A)$ can be defined as

$$\begin{aligned} r\left(\mathbf{S},\mathbf{A}\right) = {} & E_A \sum_{i}^{N_S} \delta\left[M_i = 1\right] \\ & + \sum_{i}^{N_S}\sum_{j}^{N_C} \alpha P_j n_i \Big[E_{C,E}\,\delta\left[M_i = 1\right]\delta\left[c_i^j = 1\right] \\ & + E_{C,C}\left\{\delta\left[M_i = 0\right] + \delta\left[c_i^j = 0\right] - \delta\left[M_i = 0\right]\delta\left[c_i^j = 0\right]\right\}\Big] \\ & + \sum_{i}^{N_S}\sum_{j}^{N_C} E_S\left[\delta\left[a_{i,j}^C = 0\right]\delta\left[c_i^j = 1\right] + \delta\left[a_{i,j}^C = 1\right]\delta\left[c_i^j = 0\right]\right]. \end{aligned} \tag{14}$$

5.6 Optimality equation

To minimize the expected total cost and obtain the optimal policy, we choose the expected total discount cost optimality criterion [32, 33]. Let $\nu(\mathbf{S})$ be the minimum expected total cost when the initial state is $\mathbf{S}$. Then, we can describe $\nu(\mathbf{S})$ as

$$\nu(\mathbf{S}) = \min_{\pi \in \Pi} \nu^{\pi}(\mathbf{S}) \tag{15}$$

where $\nu^{\pi}(\mathbf{S})$ is the expected total cost when the policy π with an initial state $\mathbf{S}$ is given.

The optimality equation is given by [28]

$$\nu(\mathbf{S}) = \min_{\mathbf{A} \in \mathbb{A}} \left\{ r(\mathbf{S}, \mathbf{A}) + \sum_{\mathbf{S}' \in \mathbb{S}} \lambda P[\mathbf{S}' | \mathbf{S}, \mathbf{A}] \, \nu(\mathbf{S}') \right\} \tag{16}$$

where λ is a discount factor in the MDP model. A value of λ closer to 1 gives more weight to future costs. The solution of the optimality equation corresponds to the minimum expected total cost and the optimal policy. To solve the optimality equation and to obtain the optimal policy, δ^*, we use a value iteration algorithm, as shown in Algorithm 1, which is one of the conventional algorithms to solve MDP problem, where $|\nu| = \min[\nu(\mathbf{S})]$ for $\mathbf{S} \in \mathbb{S}$ and $\nu^k(\mathbf{S})$ is the expected total cost in the step k. First, parameters are initialized (line 1 in Algorithm 1). Then, the expected total cost in the step k for each state is computed (line 2 in Algorithm 1). After that, the algorithm determines whether the iteration is sufficiently conducted or not based on the inequality $|\nu^{k+1}(\mathbf{S}) - \nu^k(\mathbf{S})| < \epsilon(1-\lambda)/2\lambda$ (line 3 in Algorithm 1), where ϵ is a stopping criterion. If the iteration is not enough, the algorithm conducts an additional iteration. Otherwise, the algorithm computes the stationary optimal policy (line 4 in Algorithm 1).

Algorithm 1 Value iteration algorithm.

1: Set $\nu^0(\mathbf{S}) = 0$ for each state $\mathbf{S}$. Specify $\epsilon > 0$, and set $k = 0$.
2: For each state $\mathbf{S}$, compute $\nu^{k+1}(\mathbf{S})$ by
$$\nu^{k+1}(\mathbf{S}) = \min_{A \in \mathbb{A}} \left\{ r(\mathbf{S}, \mathbf{A}) + \sum_{\mathbf{S}' \in \mathbb{S}} \lambda P[\mathbf{S}' | \mathbf{S}, \mathbf{A}] \, \nu^k(\mathbf{S}') \right\}$$
3: If $|\nu^{k+1}(\mathbf{S}) - \nu^k(\mathbf{S})| < \epsilon(1-\lambda)/2\lambda$, go to line 4. Otherwise, increase k by 1 and return to line 2.
4: For each state $\mathbf{S} \in \mathbb{S}$, compute the stationary optimal policy
$$\delta(\mathbf{S}) = \arg\min_{A \in \mathbb{A}} \left\{ r(\mathbf{S}, \mathbf{A}) + \sum_{\mathbf{S}' \in \mathbb{S}} \lambda P[\mathbf{S}' | \mathbf{S}, \mathbf{A}] \, \nu^{k+1}(\mathbf{S}') \right\}$$
and stop.

Generally, each iteration in the value iteration algorithm is performed in $O(|\mathbb{A}||\mathbb{S}|^2)$ [34].[10] Note the overall state space and the action space are represented by (1) and (6), respectively. Therefore, the complexity of this algorithm is decided by the number of contents, the number of SCs, the number of users, and the cache size. If these parameters have large values (i.e., state and/or action space are large), the complexity cannot be neglected even though it is a polynomial function. Therefore, the controller makes a table to store the optimal policy, which can be precomputed by the value iteration algorithm in an offline manner, and then distributes the table to SCs. This table includes the state and decision in each state. Then, SCs can decide whether to be in the sleep mode or not and which contents are stored according to the policy stored in the table.

6 Evaluation results

For performance evaluation, we compare the proposed algorithm, E3CS, with the following five schemes: (1) ADAP-SLEEP where SCs use the sleep mode only when the number of users in their area does not exceed δ_N while the contents are cached by the MDP, (2) NO-SLEEP where SCs do not use the sleep mode while the contents are cached by the MDP, (3) ALL-SLEEP where SCs are always in the sleep mode,[11] (4) HALF-SLEEP where half of SCs which are randomly selected are in the sleep mode are in the sleep mode whereas the others are in the active mode while the contents are cached by the MDP, and (5) POP-CACHE where the content is cached based on the current content popularity (not content popularity dynamics) while the mode of SCs are decided by the MDP. Note that, since this is the first work on the joint optimization of the caching strategy and sleep scheduling by considering the content popularity dynamics, we introduce simple comparison schemes to demonstrate the effectiveness of the propose scheme. The performance metric is the energy consumption ratio, ζ, based on the energy consumption of ALL-SLEEP. That is, the energy consumption of ALL-SLEEP is normalized as 1.

The default parameter settings are as follows. The number of SCs, N_S, and the total number of contents, N_C, are set to 25 and 100, respectively [35, 36]. The cache size of SCs and the average content size are assumed by 1 GB and 500 MB, respectively [36]. In particular, the content size κ_j is set to the value between 250 and 750 MB. Then, we can assume that the maximum number of contents which can be cached in SCs ξ is set to 2. Meanwhile, we assume that the link bandwidth between SC and the content server is 250 MB/s. Then, the average download completion time, $1/\mu_j$, is set to the value between 1/3 and 1. We assume that the content popularity is dynamically changed to show effectively the outperforming of the proposed scheme. The maximum number of users in one SC M_U is set to 5 [36]. The number of users in each SC is dynamically changed with the different probability. That is, the number of users in 40% of SCs is usually kept small, while the number of users in 20% of SCs is decided by a

uniform distribution. On the one hand, many users usually exist in 40% of SCs. Meanwhile, the energy consumption E_A of the active mode during τ is 6.8 W [37]. In addition, delivery energy consumptions, $E_{C,E}$ and $E_{C,C}$, are set to 0.13 and 4.2 W, respectively [37, 38].[12] The energy consumption to change contents, E_S, is set to 1 W [39].[13] On the other hand, α and τ are set to 0.1 and 1, respectively. The discount factor λ is set to 0.95. Meanwhile, we have set the x-axis values based on the real energy consumption in cellular systems [37, 38].

6.1 Effect of E_A

Figure 3 shows the effect of the energy consumption, E_A, for the active mode during τ. From Fig. 3, it can be found that the energy consumption ratio of E3CS is the lowest among the comparison schemes. This is because, in E3CS, caching strategy and sleep scheduling are jointly adjusted. In doing so, the case where the cached contents become unavailable due to the sleep mode of SCs rarely occurs. Specifically, E3CS considers the content popularity dynamics, and therefore, contents estimated to become popular are cached proactively in SCs. Therefore, the ζ of E3CS is smaller than that of POP-CACHE. Note that, in POP-CACHE, the content is cached based on the current content popularity.

As shown in Fig. 3, when E_A is smaller than 5, the energy consumption of E3CS is comparable to that of NO-SLEEP. This is because the energy consumption for the active mode is not a dominant factor to reduce the overall energy consumption. Therefore, E3CS commands all SCs to be active to reduce the energy consumption for the content delivery.

Meanwhile, as E_A increases, the energy consumption to exploit contents cached in SCs also increases. Note that SCs should be in the active mode to use contents cached in SCs. Therefore, ζ of all schemes except ALL-SLEEP increases as E_A increases. However, since E3CS commands the sleep mode adaptively depending on E_A (i.e., E3CS frequently commands the sleep mode to more SCs when E_A is large), the energy consumption ratio ζ of E3CS increases slowly when E_A is over 6.

6.2 Effect of $E_{C,C}$

The effect of the energy consumption $E_{C,C}$ to retrieve contents from the remote content server on ζ is shown in Fig. 4. From Fig. 4, it can be seen that E3CS operates adaptively even when $E_{C,C}$ is changed. Small $E_{C,C}$ means that contents can be retrieved from the remote content server without high energy consumption. In such case, E3CS commands the sleep mode to SCs to reduce the energy consumption for the active mode in SCs, and thus, the ζ of E3CS is the same as that of ALL-SLEEP when $E_{C,C}$ is 2. On the contrary, when $E_{C,C}$ has larger value (i.e., when $E_{C,C}$ is $5 \sim 6$), the energy consumption to retrieve contents from the remote content server is more influential than the energy consumption for the active mode. As a results, E3CS does not command the sleep mode to most SCs, and thus, contents can be delivered from the network edge (i.e., SCs). Therefore, the ζ of E3CS is comparable to that of NO-SLEEP.

6.3 Effect of $E_{C,E}$

The effect of the energy consumption $E_{C,E}$ to deliver contents from the network edge on ζ is shown in Fig. 5. In this result, α is set to 0.25. When $E_{C,E}$ is small (i.e., $E_{C,E} < 1$), the ζ of E3CS is almost the same as that of NO-SLEEP. This is because, when $E_{C,E}$ is small, the energy saving effect through the content delivery from the network edge is more influential than the energy consumption due to the active mode. In such a situation, SCs are always in

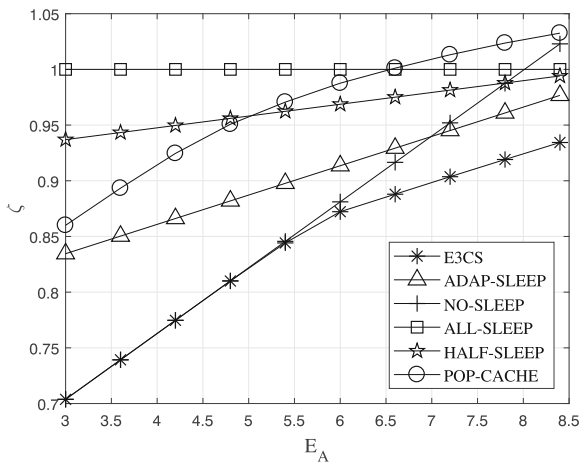

Fig. 3 Effect of the energy consumption E_A for the active mode during τ

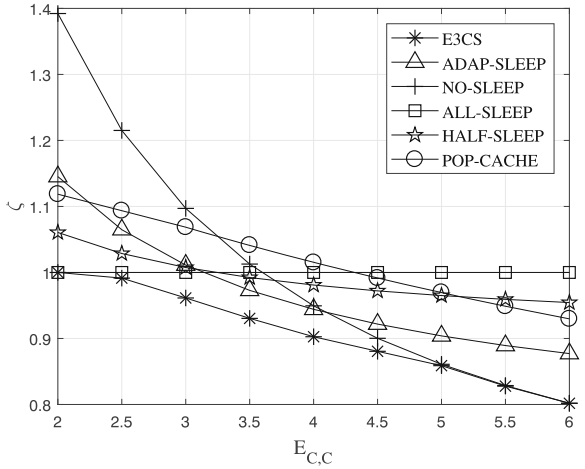

Fig. 4 Effect of the energy consumption $E_{C,C}$ to retrieve contents from the remote content server

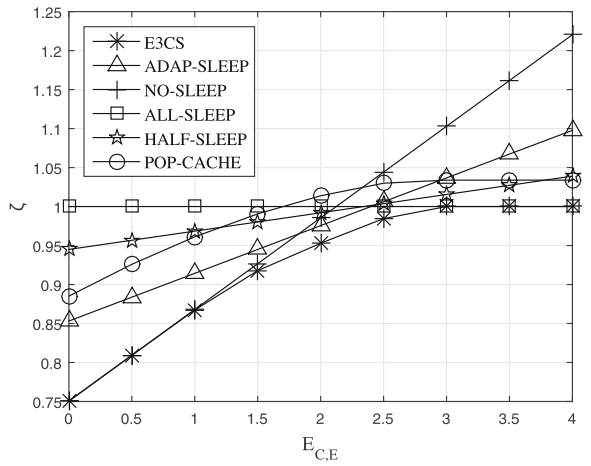

Fig. 5 Effect of the energy consumption $E_{C,E}$ to deliver contents from the network edge

the active mode to reduce the energy consumption for the content delivery. Meanwhile, as $E_{C,E}$ increases, E3CS commands the sleep mode to some SCs. Especially when $E_{C,E}$ is lager than 3, E3CS commands the sleep mode to all SCs, and therefore, the ζ of E3CS is the same as that of ALL-SLEEP. Meanwhile, since ADAP-SLEEP, NO-SLEEP, and HALF-SLEEP do not consider the energy consumption to deliver contents from the network edge, their ζ increases continuously regardless of $E_{C,E}$. Specifically, the increasing rate of NO-SLEEP is the highest, because the contents are always retrieved from the network edge in this scheme.

6.4 Effect of μ

Figure 6 shows the effect of the download rate μ between the content server and SCs. As μ increase, ζ of all schemes

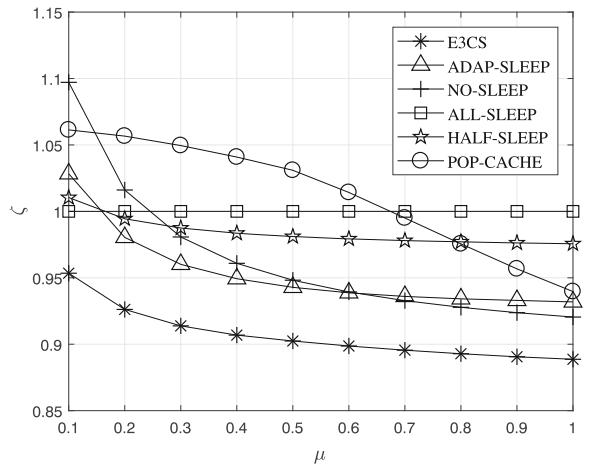

Fig. 6 Effect of the download rate μ between the content server and SCs

except ALL-SLEEP decreases. This can be explained as follows. Large μ represents that contents can be cached within short time. Then, the cached contents can be rapidly changed according to the popularity of contents, which means that the cached contents can be used more frequently. Consequently, the case where contents are delivered from the remote content server rarely occurs, and therefore, the overall energy consumption can be reduced. This result means that the bandwidth of backhaul link (i.e., the bandwidth between the content server and SCs) should be enhanced to reduce the energy consumption.

7 Conclusion

The sleep mode of small cells (SCs) can make the popular contents unavailable. To prevent this situation, in this paper, we propose an energy efficient proactive edge caching with sleep scheduling (E3CS) where the controller jointly adjusts the caching strategy and the sleep scheduling of SCs with the consideration of the content popularity dynamics. The joint optimal policy on the caching strategy and sleep scheduling minimizing the overall energy consumption is obtained by means of a Markov decision process (MDP). Evaluation results demonstrate that E3CS with the optimal policy reduces the overall energy consumption by up to 25% compared with other schemes. In addition, it can be also found that E3CS operates adaptively even when operating environments (e.g., energy consumptions for the active mode and the content delivery) are changed. In our future work, we will extend the MDP model to consider content sharing among users by means of device-to-device (D2D) communications. Moreover, to reduce the complexity of the MDP model, we will reformulate another MDP model that has an implementation-friendly threshold structure, and then evaluate the model in more practical scenarios.

Endnotes

[1] Proactive content caching is demonstrated to further boost the caching efficiency compared to reactive data caching [2–4].

[2] We assume that contents are cached in only SCs (not cached in MC) to maximize the effectiveness of edge caching. Note that the formulation is conducted based on this assumption, which can be easily extended to consider the case where contents can be cached in MC.

[3] In a long-term scenario where the candidate contents to be cached can be dynamically changed, an on-line caching algorithm needs to be considered, which is one of our future works.

[4] It is hard to define a quantitative metric to denote how frequently a new policy table should be reconstructed. Therefore, we conducted the qualitative analysis on the tradeoff between the performance improvement and the overhead depending on the frequency.

[5] Since all contents cannot be included in the state space due to large scale of contents, only popular contents (which have higher probability to be cached in small cells) should be included in the state space.

[6] This model can describe the arrival process of content requests by the time instant at which the content is requested by an user, the average number of requests, and the popularity profile (i.e., how the request rate for the content evolves over time).

[7] Note that the download completion time is inverse proportional to the size of content, κ_j. Moreover, the bandwidth of the backhaul link and background traffic volume can affect the time.

[8] Note that as the duration of decision epoch increases, the number of requests for contents during the duration of decision epoch naturally increases.

[9] Note that E_S is the average consumption during τ when SC i tries to change contents.

[10] When ϵ and λ are set to 0.001 and 0.95, respectively, the number of iterations to convergence is 65 which is not quite large. This means that the number of iterations is not a dominant factor to decide the whole running time of the value iteration algorithm.

[11] In ALL-SLEEP, since SCs are always in the sleep mode, the contents needs not to be stored.

[12] $E_{C,C}$ is set based on the link energy consumption [38].

[13] Since the energy is rarely consumed to delete data, we set E_S based on the writing operation.

Abbreviations

D2D: Device to device; E3CS: Energy efficient proactive edge caching with sleep scheduling; MC: Macro cell; MDP: Markov decision process; SC: Small cell; VOD: Video on demand

Funding

This research was supported in part by National Research Foundation (NRF) of Korea Grant funded by the Korean Government (MSIP) (No. 2017R1E1A1A01073742) and in part by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00195, Development of Core Technologies for Programmable Switch in Multi-Service Networks).

Authors' contributions

All the authors contribute to the concept, the design and developments of the methodology, and the simulation results in this manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1] Smart Quantum Communication Research Center, Korea University, Anam-ro 145, Seoul, Korea. [2] Department of Electrical and Computer Engineering, University of British Columbia, 2366 Main Mall, Vancouver, Canada. [3] School of Electrical Engineering, Korea University, Anam-ro 145, Seoul, Korea.

Received: 7 February 2018 Accepted: 17 July 2018
Published online: 02 August 2018

References

1. Cisco System, Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 (2017). https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf
2. J Tadrous, A Eryilmaz, On optimal proactive caching for mobile networks with demand uncertainties. IEEE/ACM Trans. Netw. **24**(5), 2715–2727 (2016). https://doi.org/10.1109/TNET.2015.2478476
3. V Siris, X Vasilakos, G Polyzos, in *2014 IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. Efficient proactive caching for supporting seamless mobility, (2014)
4. S Shukla, A Abouzeid, in *2017 IEEE International Conference on Computer Communications (INFOCOM)*. Proactive retention aware caching (IEEE, 2017). https://doi.org/10.1109/INFOCOM.2017.8057029
5. F Gabry, V Bioglio, I Land, On energy-efficient edge caching in heterogeneous networks. IEEE J. Sel. Areas Commun. **34**(12), 3288–3298 (2016). https://doi.org/10.1109/JSAC.2016.2611845
6. R Li, W Wang, A Huang, Z Zhang, in *2016 International Conference on Wireless Communications & Signal Processing (WCSP)*. Content caching at sleeping-enabled base stations in heterogeneous networks (IEEE, 2016). https://doi.org/10.1109/WCSP.2016.7752641
7. Y Chiang, W Liao, in *2016 IEEE International Conference on Computer Communications (INFOCOM)*. ENCORE: an energy-aware multicell cooperation in heterogeneous networks with content caching (IEEE, 2016). https://doi.org/10.1109/INFOCOM.2016.7524623
8. M Garetto, E Leonardi, S Traverso, in *2015 IEEE International Conference on Computer Communications (INFOCOM)*. Efficient analysis of caching strategies under Dynamic content popularity (IEEE, 2015). https://doi.org/10.1109/INFOCOM.2015.7218613
9. R Crane, D Sornette, Robust dynamic classes revealed by measuring the response function of a social system. Proc. Natl. Acad. Sci. U. S. A. **105**(41), 15649–15653 (2008). https://doi.org/10.1073/pnas.0803685105
10. J Yang, J Leskovec, in *2011 ACM International Conference on Web Search and Data Minig (WSDM)*. Patterns of temporal variation in online media (ACM, 2011). https://doi.org/10.1145/1935826.1935863
11. Y Zhou, L Chen, C Yang, D Chiu, Video popularity dynamics and its implication for replication. IEEE Trans. Multimed. **17**(8), 1273–1285 (2015). https://doi.org/10.1109/TMM.2015.2447277
12. G Szabo, B Huberman, Predicting the popularity of online content. Commun. ACM. **53**(8), 80–88 (2010). https://doi.org/10.1145/1787234.1787254
13. C Li, J Liu, S Ouyang, Characterizing and predicting the popularity of online videos. IEEE Access. **4**, 1630–1641 (2016). https://doi.org/10.1109/ACCESS.2016.2552218
14. S Traverso, M Ahmed, M Garetto, P Giaccone, E Leonardi, S Niccolini, Temporal locality in today's content caching: why it matters and how to model it. ACM SIGCOMM Comput. Commun. Rev. **43**(5), 5–12 (2013). https://doi.org/10.1145/2541468.2541470
15. S He, H Tian, X Lyu, Edge popularity prediction based on social-driven propagation dynamics. IEEE Commun. Lett. **21**(5), 1027–1030 (2017). https://doi.org/10.1109/LCOMM.2017.2655038
16. S Nikolaou, R Van Renesse, N Schiper, Proactive cache placement on cooperative client caches for online social networks. IEEE Trans. Parallel Distrib. Syst. **27**(4), 1174–1186 (2016). https://doi.org/10.1109/TPDS.2015.2425398
17. X Wang, S Leng, K Yang, Social-aware edge caching in fog radio access networks. IEEE Access. **5**, 8492–8501 (2017). https://doi.org/10.1109/ACCESS.2017.2693440

18. J Kakar, S Gherekhloo, ZH Awan, A Sezgin, in *2017 IEEE International Conference on Communications (ICC)*. Fundamental limits on latency in cloud- and cache-aided hetnets (IEEE, 2017). https://doi.org/10.1109/ICC.2017.7996346
19. MA Maddah-Ali, U Niesen, Fundamental limits of caching. IEEE Trans. Inf. Theory. **60**(5), 2856–2867 (2014). https://doi.org/10.1109/TIT.2014.2306938
20. Y Fadlallah, A Tulino, D Barone, G Vettigli, J Llorca, J Gorce, Coding for caching in 5G networks. IEEE Commun. Mag. **55**(2), 106–113 (2017). https://doi.org/10.1109/MCOM.2017.1600449CM
21. K Poularakis, G Iosifidis, A Argyriou, I Koutsopoulos, L Tassiulas, in *2016 IEEE International Conference on Computer Communications (INFOCOM)*. Caching and operator cooperation policies for layered video content delivery (IEEE, 2016). https://doi.org/10.1109/INFOCOM.2016.7524427
22. Y Zhou, Z Zhao, R Li, H Zhang, Y Louet, Cooperation based probabilistic caching strategy in clustered cellular networks. IEEE Commun. Lett. **21**(9), 2029–2032 (2017). https://doi.org/10.1109/LCOMM.2017.2717398
23. S Krishnan, M Afshang, H Dhillon, Effect of retransmissions on optimal caching in cache-enabled small cell networks. IEEE Trans. Veh. Technol. **66**(12), 11383–11387 (2017). https://doi.org/10.1109/TVT.2017.2721839
24. S Muller, O Atan, M Schaar, A Klein, Context-aware proactive content caching with service differentiation in wireless networks. IEEE Trans. Wirel. Commun. **16**(2), 1024–1036 (2017). https://doi.org/10.1109/TWC.2016.2636139
25. D Liu, C Yang, Energy efficiency of downlink networks with caching at base stations. IEEE J. Sel. Areas Commun. **34**(4), 907–922 (2016). https://doi.org/10.1109/JSAC.2016.2549398
26. B Perabathini, E Bastug, M Kountouris, M Debbah, A Contey, in *2015 IEEE International Conference on Communications (ICC) Workshop*. Caching at the edge: a green perspective for 5G networks (IEEE, 2015). https://doi.org/10.1109/ICCW.2015.7247608
27. H Ko, G Lee, D Suh, S Pack, X Shen, An optimized and distributed data packet forwarding in LTE/LTE-A networks. IEEE Trans. Veh. Technol. **65**(5), 3462–3473 (2016). https://doi.org/10.1109/TVT.2015.2432152
28. M Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. (Wiley, Hoboken, 1994)
29. E Feinberg, A Shwartz, *Handbook of Markov Decision Processes: Methods and Applications*. (Kluwer Academic Publishers, The Netherlands, 2002)
30. S Singh, H Dhillon, J Andrews, Offloading in heterogeneous networks: modeling, analysis, and design insights. IEEE Trans. Wirel. Commun. **12**(5), 2484–2497 (2013). https://doi.org/10.1109/TWC.2013.040413.121174
31. H Ko, J Lee, S Pack, MALM: mobility-aware location management scheme in femto/macrocell networks. IEEE Trans. Mob. Comput. **16**(11), 3115–3125 (2017). https://doi.org/10.1109/TMC.2017.2690287
32. J Pan, W Zhang, in *2012 IEEE Vehicular Technology Conference (VTC Fall)*. An MDP-based handover decision algorithm in hierarchical LTE networks (IEEE, 2012). https://doi.org/10.1109/VTCFall.2012.6398908
33. H Tabrizi, G Farhadi, J Cioffi, in *2011 IEEE Global Telecommunications Conference (GLOBECOM)*. A learning-based network selection method in heterogeneous wireless systems (IEEE, 2011). https://doi.org/10.1109/GLOCOM.2011.6134269
34. M Littman, T Dean, L Kaelbling, in *1995 International Conference on Uncertainty in Artificial Intelligence (UAI)*. On the complexity of solving markov decision problems (UAI, 1995). https://dl.acm.org/citation.cfm?id=2074203
35. M Chen, Y Hao, L Hu, K Huang, V Lau, Green and mobility-aware caching in 5G networks. IEEE Trans. Wirel. Commun. **16**(12), 8347–8361 (2017). https://doi.org/10.1109/TWC.2017.2760830
36. C Zhan, Z Wen, Content cache placement for scalable video in heterogeneous wireless network. IEEE Commun. Lett. **21**(12), 2714–2717 (2017). https://doi.org/10.1109/LCOMM.2017.2756033
37. S Zhang, N Zhang, S Zhou, J Gong, Z Niu, X Shen, Energy-aware traffic offloading for green heterogeneous networks. IEEE J. Sel. Areas Commun. **34**(5), 1116–1129 (2016). https://doi.org/10.1109/JSAC.2016.2520244
38. A Bianco, R Mashayekhi, M Meo, in *2016 IEEE International Conference on Communications (ICC)*. Energy consumption for data distribution in content delivery networks (IEEE, 2016). https://doi.org/10.1109/ICC.2016.7511356
39. J Zedlewski, S Sobti, N Garg, F Zheng, A Krishnamurthy, R Wang, in *2003 USENIX Conference on File and Storage Technologies*. Modeling Hard-Disk Power Consumption (USENIX, 2003). https://www.usenix.org/legacy/event/fast03/tech/full_papers/zedlewski/zedlewski_html/