On-Bound Selection Cache Replacement Policy for Wireless Data Access

Hui Chen, *Member*, *IEEE*, and Yang Xiao, *Senior Member*, *IEEE*

Abstract—Cache can be used for mobile devices to reduce the usage of limited bandwidth in wireless networks. Ideally, frequently accessed and infrequently updated data items should be cached and infrequently accessed and frequently updated data items should be evicted or not cached at all. Most of the existing cache replacement policies adopt only access information so that frequently updated data items are also cached. As a remedy, we propose a cache replacement policy, called On-Bound Selection (OBS), that uses both data access and update information. The proposed OBS is inspired by an analytical analysis for a server-based Poll-Each-Read (SB-PER) and a revised Call-Back (R-CB). The OBS provides an upper bound for effective hit ratio and a lower bound for communication cost. The proposed scheme is evaluated and compared with a least frequently used (LFU) replacement policy through extensive simulations. Simulation results show that the OBS outperforms LFU in terms of both effective hit ratio and communication cost.

Index Terms—Cache, replacement policy, wireless networks, access, update, effective hit ratio, communication cost.

1 Introduction

Mobile Terminals (MTs) have been used to access information stored in remote servers through wireless communication channels. However, wireless channels are the scarcest and most expensive resource in entire networks. Cache mechanisms have been introduced to reduce the bandwidth usage of wireless channels [3], [7], [8], [13], [15], [20], [27], [29]. In general, a cache mechanism consists of a cache access algorithm and a replacement policy. A cache access algorithm defines how a client and its server exchange messages and data to achieve a certain degree of data consistency. Examples of cache access algorithms are Poll-Each-Read (PER) [20], [21], Call-Back (CB) [14], [20], [31], and Invalidation Report (IR) [3]. Many applications use strong consistent cache access algorithm [3], [7], [8] to prevent using stalled data. A replacement policy decides which cached data item is evicted when an uncached data item is accessed and the cache is full. Replacement policies affect the overall cache performance. They are more important to wireless MTs than to wired terminals since wireless terminals generally have less cache space than wired ones.

Many cache replacement policies have been proposed for wired and wireless networks [2], [26], [29], [30]. However, most of these replacement policies use only data access information and ignore data update information. Update information is critical and should not be ignored because an update makes a cached data item invalid and a cache hit becomes useless. Two replacement policies [29], [30] have been proposed to use both access and update information with IR schemes. However, their design goal is to reduce the stretch, a normalized delay, which is not the scarce and expensive resource in entire networks and is, at least sometimes, not the best design goal. Furthermore, as pointed out in [20], IR schemes require broadcasting in entire networks and are not suitable for implementation in realistic wireless networks. IR schemes may also require low layer functions (for example, data link layer) of wireless network protocols, which may not be available for implementing caching in upper layer applications (for example, application layer).

This paper studies cache replacement policies. Our goal is to design a replacement policy for increasing effective cache hit ratio and decreasing communication cost as much as possible. Two update-based strong consistent cache access algorithms, a server-based PER (SB-PER) and a revised CB (R-CB), are introduced and they provide both access and update histories to replacement policies. They can also be run in application layers and do not rely on lower layer functionalities of wireless network protocols. Intuitively, a good replacement policy should evict infrequently accessed but frequently updated data items. We analytically analyze access and update processes in the SB-PER and the R-CB. The analysis provides an upper bound of effective hit ratio and a lower bound of communication cost. The analytical analysis inspires us with how infrequently accessed but frequently updated data items can be chosen. We hence design a replacement policy, called *On-Bound Selection (OBS)*, to evict infrequently accessed but frequently updated data items while keeping frequently accessed but infrequently updated data items according to the defined bounds. Our proposed *OBS* replacement policy is a frequency-based replacement policy. It is not limited to wireless data access and it is also applicable to wired networks such as Internet client-server applications and Web caching.

The rest of this paper is organized as follows: Section 2 shows the network architecture, the data access model, and

- *H. Chen is with the Department of Mathematics and Computer Science, Virginia State University, Petersburg, VA 23806. E-mail: huichen@ieee.org.*
- *Y. Xiao is with the Department of Computer Science, University of Alabama, 101 Houser Hall, Box 870290, Tuscaloosa, AL 35487-0290. E-mail: yangxiao@ieee.org.*

Manuscript received 22 Jan. 2006; revised 17 Oct. 2006; accepted 13 June 2007; published online 9 July 2007.
Recommended for acceptance by A. Mei.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0023-0106.
Digital Object Identifier no. 10.1109/TC.2007.70768.

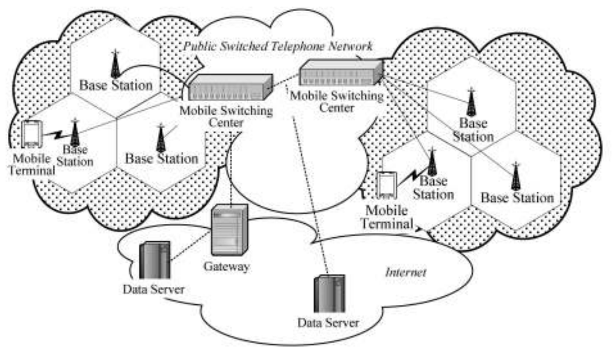

Fig. 1. Wireless network architecture.

several examples of wireless data applications. The SB-PER and the R-CB access algorithms are introduced in Section 3. We propose the update-based replacement policy in Section 4. The analytical model of cache access and update is presented in Section 5. In Section 6, we briefly present our motivation and comments on cache replacement policies. Performance evaluation and comparison are given in Section 7. We conclude the paper in Section 8.

2 Network Architecture and Applications

2.1 Network Architecture Overview

Wireless data applications and services have been provided on personal communication service (PCS) networks [25]. In current PCS networks, a service area is divided into a number of location areas (LAs). An LA is further partitioned into a number of cells. Each cell has a base station (BS), which controls many MTs within the cell via wireless links. All of the BSs within one LA are connected to a mobile switching center (MSC). All of the MSCs are finally connected to a public switched telephone network (PSTN). A PCS can be connected to the Internet in many ways, such as wireless carriers' proprietary networks and PSTN. An MT can access data servers residing in either PCS networks or the Internet via its corresponding BS. Fig. 1 shows an example of a wireless network architecture for wireless data access.

In general, as shown in Fig. 1, wireless data access follows client-server models. Databases are hosted at remote servers, which are usually in the wired networks. In this paper, we do not distinguish among clients, MTs, and users. A client accesses data items in a database in a remote server through wireless links when the data item is not in cache or is cached but not valid. The server updates data items when requested by clients or others.

2.2 Wireless Data Applications

We briefly show several examples of wireless network applications which can benefit from the proposed cache access algorithms and replacement policies. Wireless Web applications are particular examples of such applications. Many network protocols can be used to support wireless network data applications.

The Wireless Application Protocol (WAP) is designed to enable easy, fast delivery of relevant information and services to mobile users in wireless cellular networks. WAP applications run in client-server models. Cache operations have been proposed for WAP applications [18].

In [20], a business card application is proposed for iSMS, a platform that integrates the IP network with the Short Message Services (SMS) in a mobile telephone network. A subscriber can store a phone book consisting of a number of business cards in a remote server. This server-based application has many advantages. First, the subscriber can access the phone book through different MTs. Second, the application server may provide directory services such as yellow pages and white pages.

In [16], an online auction Web application is proposed. An auction item is associated with its description, price, and reviews. Some of the items change more frequently than others do. For example, price may change every few minutes and the description remains the same during the auction. This application is not necessarily implemented as a Web application and it is beneficial to implement this application on wireless networks because a user may have a desire to start an auction anytime and anywhere.

A wireless application called a real-time stock price monitor can be used to check stock prices and a user can make a transaction by using the MT based on the stock information obtained from a server. In this application, the data sources may or may not be located outside the cellular networks. Stocks are updated and accessed at different rates. Some stocks may be traded in a huge volume. Their prices may change very frequently.

These examples show that update and access frequencies can be different among different data items. In fact, as shown in [5], many data items stored in remote databases and accessed through Web applications change frequently. Therefore, it is important to accommodate both update and access information into cache mechanism design. Furthermore, it is generally impossible to cache all server-kept data items such as the business cards of all subscribers, the prices of all auction merchandise, and the prices of all stocks in local client caches. Inevitably, cache replacement policies need to be taken into account. Nevertheless, as we will discuss in Section 7.5, it is not necessarily good for the overall system performance to maximize client cache space in networks with updates, especially heavy updates. Consequently, replacement policies are better as an integral part of the network system architecture.

3 Cache Access Algorithms

PER and CB are two widely used fundamental cache access algorithms for a client-server data access model [20], [21]. In PER, a client attempts to read a data item from a server at each access and the server only replies to it with an acknowledgment when the data item is cached and valid, where a valid data item means that the data item has not been updated since it was cached, and, if the data item is invalid, the server sends the data item to the client. In CB, the server sends an invalidation message at each update to the client, which caches the updated data item. The details of these two schemes can be found in [20]. In this section, we introduce an SB-PER and an R-CB suitable for using both update and access information in replacement policies. Both of these cache access algorithms have the capability of

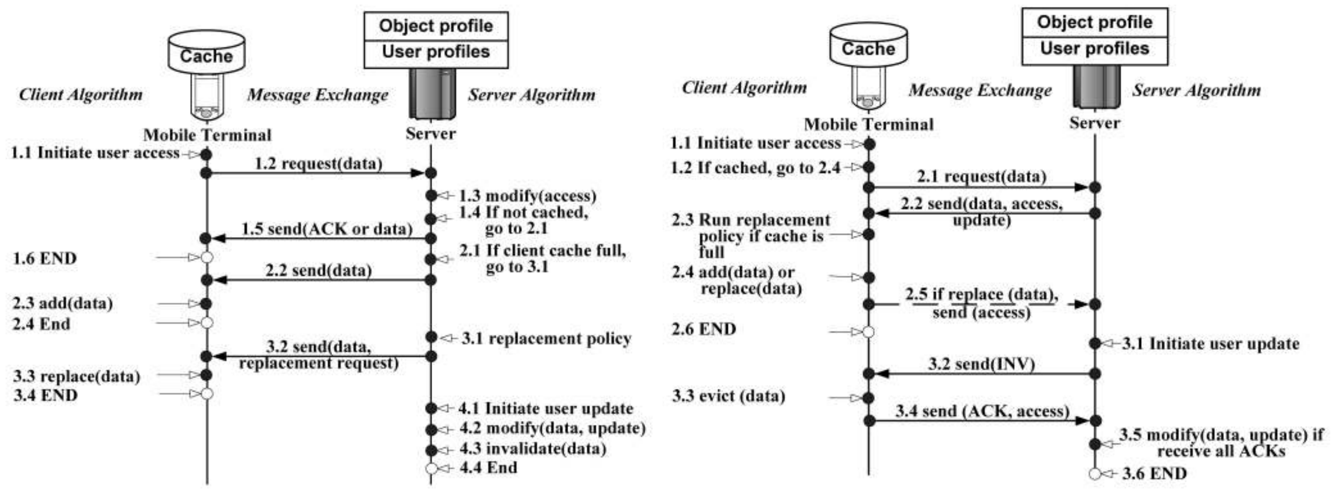

Fig. 2. (a) The SB-PER and (b) the R-CB.

maintaining full access and update histories and thus can use any replacement policy using both data access and update information.

3.1 Server-Based Poll-Each-Read

As shown in Fig. 2a, the SB-PER uses a stateful server, which stores access (Step 1.3) and updates (Step 4.2) the histories of all data items. The server becomes the most knowledgeable entity in the network and, thus, it is capable of making the wisest decisions. Therefore, the server makes replacement decisions for a client (Step 3.1) and sends decisions to the client (Step 3.2). At the server, an object profile is used for storing the update histories of all data items.

3.2 Revised Call-Back

As shown in Fig. 2b, we revised the original CB algorithm such that the server records the access and update histories of all data items (Steps 2.5 and 3.5) and the access and update histories of the cached data items at clients are synchronized through the messages. We shall call it the R-CB. Note that the server saves the access history of a cached data item by sending a message to the server (Step 2.5) when it is evicted to accommodate the accessed data item. The access and update histories of a data item are sent to a client with the data item by the server (Step 2.2). Therefore, the client has the up-to-date access and update histories of all of the cached and accessed data items and, thus, it can make the wisest replacement decision (Step 2.3). As shown in Fig. 2b, the user profiles and the object profile have the same usage as the SB-PER.

4 On-Bound Selection Replacement

In this section, we propose an update-based cache replacement policy, called *OBS*, which uses both update and access frequencies. The OBS tries to cache frequently accessed but infrequently updated data items and to evict infrequently accessed but frequently updated data items in the cache. In other words, the OBS tries to *keep good data items* and *evict bad data items* in the cache.

Let $f_{ij}^a(t)$ denote the access frequency of data item O_j at client i and let $f_j^u(t)$ denote the update frequency of data item O_j at the server up to time t, respectively. Note that $f_j^u(t)$ does not have a subscript i, but $f_{ij}^a(t)$ does since $f_j^u(t)$ is a global statistic. Denote μ_{ij} and λ_j as the access rate of data item O_j at the client i and the update rate at the server, respectively. We expect that $f_{ij}^a(t)$ and $f_j^u(t)$ approach the access rate μ_{ij} and the update rate λ_j, respectively, when time t is significantly large, that is, $\mu_{ij} = \lim_{t\to\infty} f_{ij}^a(t)$ and $\lambda_j = \lim_{t\to\infty} f_j^u(t)$. We define an "*On-Bound Selection*" factor (OBS factor) as follows:

$$OBSF_{ij}(t) = \frac{\left(f_{ij}^a(t)\right)^2}{f_{ij}^a(t) + f_j^u(t)}, \tag{1}$$

$$OBSF_{ij} = \lim_{t\to\infty} OBSF_{ij}(t) = \frac{(\mu_{ij})^2}{\mu_{ij} + \lambda_j}. \tag{2}$$

In the OBS replacement policy, the cache access algorithms maintain access and update histories, which can be used to compute the OBS factor defined in (1). The replacement policy computes the OBS factors of all of the cache data items and the newly accessed data item and then looks for the data item with the least OBS factor. If the data item with the least OBS factor is not the newly accessed data item, the corresponding data item is replaced with the newly accessed data item; otherwise, the accessed data item is not cached and there is no change in the cache.

The OBS policy is a frequency-based replacement policy since it uses access and update frequencies. First, as we have discussed in Section 3, the SB-PER and the R-CB are the cache access algorithms that are capable of maintaining access and update histories of all data items. Therefore, the OBS policy can be applied to the SB-PER and the R-CB. Second, the access and update frequencies are measured by using a moving window. For the rest of the paper, the performance of the OBS policy is studied for these two cache access algorithms. However, the proposed replacement policy should not be limited to just these two cache access algorithms and it should be suitable for any cache access algorithm that can provide both access and update frequencies.

5 Analytical Model

In this section, we provide an analytical analysis that gives the reasoning behind the OBS scheme. The network has a data server and many clients. All data accesses happen at clients and all updates happen at the server. We have the following assumptions:

1. The server has N data items.
2. A client has a cache that holds up to K data items.
3. Accesses to data item O_i follow a Poisson process with rate μ_i.
4. Updates to data item O_i follow a Poisson process with rate λ_i.

An effective cache hit is defined as an event where an access happens to a data item that is cached and valid. The effective hit ratio of a cache access algorithm with a replacement policy is the probability that an access causes an effective cache hit when the cache access algorithm is exercised with the replacement policy. Let p_{SB-PER} and p_{R-CB} denote the effective hit ratios of the SB-PER and the R-CB with a given replacement policy, respectively.

Define the effective hit ratio of data item O_i of a cache access algorithm with a replacement policy as the probability that an access causes an effective cache hit and the accessed data item is O_i when the cache access algorithm is exercised with a replacement policy. Let $p_{SB-PER,i}$ and $p_{R-CB,i}$ denote the effective hit ratios of data item O_i for the SB-PER and the R-CB with a given replacement policy, respectively.

Since wireless channels between MTs and their BSs are the scarcest and most expensive resources, we only consider the bandwidth usage of messages and data transmitted between MTs and their corresponding BSs. Define communication cost as the average bytes transferred through wireless links between MTs and the BSs per data access. Let c_{SB-PER} and c_{R-CB} denote communication costs for the SB-PER and the R-CB with a given replacement policy, respectively.

Let X_i denote the random variable of the number of accesses to data item O_i occurring between two consecutive updates to data item O_i. Let $p(X_i = k)$ denote the probability that exactly k accesses to data item O_i occurring between two consecutive updates to data item O_i. $p(X_i = k)$ is independent of replacement polices. Let $E(X_i)$ denote the average number of accesses to data item O_i occurring between two consecutive updates to data item O_i. $E(X_i)$ depends on both access and update processes, but is independent of replacement policies. Let $E(X_{SB-PER,i})$ and $E(X_{R-CB,i})$ denote the average numbers of effective cache hits to data item O_i between two consecutive updates to the same data item when the SB-PER or the R-CB is exercised with a given replacement policy, respectively. Let $n^U_{SB-PER,i}$ and $n^U_{R-CB,i}$ be upper bounds of the average numbers of effective cache hits to data item O_i between two consecutive updates to the same data item when the SB-PER and R-CB are exercised, respectively, with a given replacement policy.

The message transmission costs for a request message (Step 1.2 in Fig. 2a and Step 2.1 in Fig. 2b), an acknowledge message (Step 1.5 in Fig. 2a and Step 3.4 in Fig. 2b), and an invalidation message (Step 3.2 in Fig. 2b) are c_{req}, c_{ack}, and c_{inv}, respectively. Assume that the cost for transmitting any data item is the same and is denoted as c_{obj}, that is, we assume that the data items are of the same size. We assume that message headers have some extra fields that can be used for holding access and update frequency information and replacement decisions. Therefore, the access/update frequency information in messages in Step 3.2 in Fig. 2a and Steps 2.2 and 3.4 in Fig. 2b is not counted, only the message in Step 2.5 of Fig. 2b is counted and its cost is denoted as c_a.

Lemma 1. *The following inequalities always hold for both the SB-PER and the R-CB with any replacement policy:*

$$p_{SB-PER,i} \le \frac{1}{\mu}\frac{\mu_i^2}{\lambda_i + \mu_i}, \tag{3}$$

$$p_{R-CB,i} \le \frac{1}{\mu}\frac{\mu_i^2}{\lambda_i + \mu_i}. \tag{4}$$

Proof. According to [28],

$$p(X_i = k) = \frac{\lambda_i}{\lambda_i + \mu_i}\left(\frac{\mu_i}{\lambda_i + \mu_i}\right)^k, \tag{5}$$

$$E(X_i) = \sum_{k=0}^{\infty} kp(X_i = k) = \frac{\mu_i}{\lambda_i}. \tag{6}$$

There are three cases in total: 1) There is no access to data O_i between two consecutive updates to O_i, 2) there is only one access to data O_i between two consecutive updates to O_i, and 3) there are $n \ge 2$ accesses to data O_i between two consecutive updates. When an access to data item O_i happens, O_i may be cached if it is the first access between two consecutive updates to O_i and, then, subsequent accesses can be cache hits. Note that replacements make the number of cache hits fewer than the number of subsequent accesses, that is, cases 1 and 2 do not contribute to any effective cache hit definitely and only case 3 contributes at most $n-1$ effective cache hits no matter which replacement policy is used. Therefore, regarding data item O_i, from (5), we can derive an upper bound of the number of effective cache hits to data item O_i between two consecutive updates to data item O_i for the SB-PER for a given replacement policy, that is,

$$\begin{aligned} n^U_{SB-PER,i} &= \sum_{k=2}^{\infty}(k-1)p(X_i = k) \\ &= \sum_{k=2}^{\infty}(k-1)\frac{\lambda_i}{\lambda_i + \mu_i}\left(\frac{\mu_i}{\lambda_i + \mu_i}\right)^k = \frac{\lambda_i \mu_i^2}{\lambda_i + \mu_i} \end{aligned} \tag{7}$$

and

$$E(X_{SB-PER,i}) \le n^U_{SB-PER,i} \le E(X_i). \tag{8}$$

Because access and update are two Poisson processes, according to (6), (7), and (8),

$$\begin{aligned} p_{SB-PER,i} &= \frac{\mu_i}{\mu}\frac{E(X_{SB-PER,i})}{E(X_i)} \le \frac{\mu_i}{\mu}\frac{n^U_{SB-PER,i}}{E(X_i)} \\ &= \frac{\mu_i}{\mu}\frac{\lambda_i\mu_i^2}{\lambda_i+\mu_i}\Big/\frac{\mu_i}{\lambda_i} = \frac{1}{\mu}\frac{\mu_i^2}{\lambda_i+\mu_i}. \end{aligned} \tag{9}$$

Similarly, we have

$$p_{R-CB,i} = \frac{\mu_i}{\mu}\frac{E(X_{R-CB,i})}{E(X_i)} \le \frac{\mu_i}{\mu}\frac{n^U_{R-CB,i}}{E(X_i)} = \frac{1}{\mu}\frac{\mu_i^2}{\lambda_i+\mu_i}. \tag{10}$$

□

Theorem 1. *The following inequalities always hold for both the SB-PER and the R-CB with any replacement policy:*

$$p_{SB-PER} \le \frac{1}{\mu}\sum_{i=1}^{N}\frac{\mu_i^2}{\lambda_i+\mu_i}, \tag{11}$$

$$p_{R-CB} \le \frac{1}{\mu}\sum_{i=1}^{N}\frac{\mu_i^2}{\lambda_i+\mu_i}. \tag{12}$$

Proof. According to Lemma 1, when the SB-PER is exercised with a given replacement policy, we have

$$p_{SB-PER} = \sum_{i=1}^{N} p_{SB-PER,i} \le \sum_{i=1}^{N}\frac{1}{\mu}\frac{\mu_i^2}{\lambda_i+\mu_i} = \frac{1}{\mu}\sum_{i=1}^{N}\frac{\mu_i^2}{\lambda_i+\mu_i}. \tag{13}$$

Similarly,

$$p_{R-CB} = \sum_{i=1}^{N} p_{R-CB,i} \le \sum_{i=1}^{N}\frac{1}{\mu}\frac{\mu_i^2}{\lambda_i+\mu_i} = \frac{1}{\mu}\sum_{i=1}^{N}\frac{\mu_i^2}{\lambda_i+\mu_i}. \tag{14}$$

□

Theorem 1 indicates that we have found an upper bound for the effective hit ratio of the SB-PER and an upper bound for the effective ratio of the R-CB. Denote them as p^U_{SB-PER} and p^U_{R-CB}, respectively, that is,

$$p^U_{SB-PER} = p^U_{R-CB} = \frac{1}{\mu}\sum_{i=1}^{N}\frac{\mu_i^2}{\lambda_i+\mu_i}.$$

Corollary 1. *The following inequality always holds for the SB-PER with any replacement policy:*

$$c_{SB-PER} \ge c_{req} + \left(1 - p^U_{SB-PER}\right)c_{obj} + p^U_{SB-PER}c_{ack}. \tag{15}$$

Proof. According to our algorithm illustrated in Fig. 1a, the communication cost of the SB-PER with a given replacement policy is

$$c_{SB-PER} = c_{req} + (1 - p_{SB-PER})c_{obj} + p_{SB-PER}c_{ack}. \tag{16}$$

Then,

$$\frac{dc_{SB-PER}}{dp_{SB-PER}} = c_{ack} - c_{obj}. \tag{17}$$

Without loss of generality, assume that $c_{ack} < c_{obj}$. Then,

$$\frac{dc_{SB-PER}}{dp_{SB-PER}} < 0. \tag{18}$$

Equation (18) indicates that c_{SB-PER} is a monotonically decreasing function of p_{SB-PER}. Since p^U_{SB-PER} is an upper bound, $p_{SB-PER} \le p^U_{SB-PER}$. Therefore,

$$c_{SB-PER} \ge c_{req} + \left(1 - p^U_{SB-PER}\right)c_{obj} + p^U_{SB-PER}c_{ack}. \tag{19}$$

□

Corollary 2. *The following inequality always holds for the R-CB with any replacement policy:*

$$c_{R-CB} \le \left(1 - p^U_{R-CB}\right)\left(c_{req} + c_{obj}\right). \tag{20}$$

Proof. According to our algorithm illustrated in Fig. 1b, the communication cost of the R-CB is

$$c_{R-CB} = (1 - p_{R-CB})\left(c_{req} + c_{obj}\right) + p_{inv}(c_{inv} + c_{ack}) + p_{rep}c_a, \tag{21}$$

$$\frac{dc_{R-CB}}{dp_{R-CB}} = -\left(c_{req} + c_{obj}\right) < 0, \tag{22}$$

$$\frac{dc_{R-CB}}{dp_{inv}} = (c_{inv} + c_{ack}) > 0, \tag{23}$$

$$\frac{dc_{R-CB}}{dp_{rep}} = c_a > 0. \tag{24}$$

Equations (22), (23), and (24) indicate that c_{R-CB} is a monotonically decreasing function of p_{R-CB}, a monotonically increasing function of p_{inv}, and a monotonically increasing function of p_{rep}, respectively. We know that $p_{inv} \ge 0$ and $p_{rep} \ge 0$. Given that p^U_{R-CB} is an upper bound of the effective hit ratio, that is, $p_{R-CB} \le p^U_{R-CB}$, we have

$$\begin{aligned} c_{R-CB} &= (1 - p_{R-CB})\left(c_{req} + c_{obj}\right) + p_{inv}(c_{inv} + c_{ack}) + p_{rep}c_a \\ &\le (1 - p_{R-CB})\left(c_{req} + c_{obj}\right) + p_{inv}(c_{inv} + c_{ack}) + 0 \cdot c_a \\ &\le (1 - p_{R-CB})\left(c_{req} + c_{obj}\right) + 0 \cdot (c_{inv} + c_{ack}) \\ &\le (1 - p_{R-CB})\left(c_{req} + c_{obj}\right) \le \left(1 - p^U_{R-CB}\right)\left(c_{req} + c_{obj}\right). \end{aligned} \tag{25}$$

□

Corollaries 1 and 2 indicate that we have found a lower bound for the communication cost of the SB-PER and a lower bound for the communication cost of the R-CB. Denote them as c^L_{SB-PER} and c^L_{R-CB}, respectively, that is,

$$c^L_{SB-PER} = c_{req} + \left(1 - p^U_{SB-PER}\right)c_{obj} + p^U_{SB-PER}c_{ack}$$

and $c^L_{R-CB} = (1 - p^U_{R-CB})(c_{req} + c_{obj})$.

Theorem 2. *The SB-PER and the R-CB (with any replacement policy) reach their effective hit ratio upper bounds p^U_{SB-PER} and p^U_{R-CB} given in Theorem 1, respectively, when $K \ge N$.*

Proof. Since $K \ge N$, there is no replacement. Without loss of generality, assume that the cache is divided into N slots, and each slot can hold a data item. Associate each data item with a particular slot. A data item is always cached at a certain slot. Therefore, caching different data items becomes disjoint events. We have

$$p_{SB-PER} = p_{R-CB} = \sum_{i=1}^{N} \frac{\mu_i}{\mu} \frac{\mu_i}{\lambda_i + \mu_i} = \frac{1}{\mu} \sum_{i=1}^{N} \frac{\mu_i^2}{\lambda_i + \mu_i}, \tag{26}$$

that is,

$$p_{SB-PER} = p_{R-CB} = p_{SB-PER}^U = p_{R-CB}^U. \tag{27}$$

□

Corollary 3. *The SB-PER (with any replacement policy) reaches its communication cost lower bound c_{SB-PER}^L given in Corollary 1 when $K \geq N$.*

Proof. According to Corollary 2 and Theorem 2,

$$\begin{aligned} c_{SB-PER}^L &= c_{req} + \left(1 - p_{SB-PER}^U\right)c_{obj} + p_{SB-PER}^U c_{ack} \\ &= c_{req} + (1 - p_{SB-PER})c_{obj} + p_{SB-PER}c_{ack} = c_{SB-PER}. \end{aligned} \tag{28}$$

□

Corollary 4. *When $K \geq N$, the communication cost of the R-CB (with any replacement policy) is*

$$c_{R-CB} = (1 - p_{R-CB})\left(c_{req} + c_{obj} + c_{inv} + c_{ack}\right), \tag{29}$$

which is larger than c_{R-CB}^L, given in Corollary 2 with no more than two message costs.

Proof. Since $K \geq N$, there is no replacement, that is, $p_{rep} = 0$. We also have

$$p_{inv} = \sum_{i=1}^{N} \frac{\mu_i}{\mu} \frac{\lambda_i}{\lambda_i + \mu_i} = \frac{1}{\mu} \sum_{i=1}^{N} \frac{\lambda_i \mu_i}{\mu_i + \lambda_i} = 1 - p_{R-CB}^L, \tag{30}$$

$$c_{R-CB} = (1 - p_{R-CB})\left(c_{req} + c_{obj}\right) + p_{inv}(c_{inv} + c_{ack}) + p_{rep}c_a. \tag{31}$$

Therefore, when $K \geq N$, we have

$$c_{R-CB} = \left(1 - p_{R-CB}^L\right)\left(c_{req} + c_{obj} + c_{inv} + c_{ack}\right). \tag{32}$$

According to Corollary 2 and Theorem 2,

$$\begin{aligned} c_{R-CB} &= \left(1 - p_{R-CB}^U\right)\left(c_{req} + c_{obj} + c_{inv} + c_{ack}\right) \\ &\geq \left(1 - p_{R-CB}^U\right)\left(c_{req} + c_{obj}\right) = c_{R-CB}^L. \end{aligned} \tag{33}$$

□

5.1 Remarks on the OBS Replacement Policy

According to the definition of the OBS factor and Lemma 1, we have the following observations about the effective hit ratios of data item O_i for the SB-PER and the R-CB, respectively, at client k:

$$\begin{aligned} p_{SB-PER,i}^U = p_{R-CB,i}^U &= \frac{1}{\mu} \frac{\mu_{ki}^2}{\lambda_i + \mu_{ki}} \\ &= \lim_{t \to \infty} \frac{1}{\mu} \frac{\left(f_{ki}^a(t)\right)^2}{f_{ki}^a(t) + f_i^u(t)} = \frac{1}{\mu} \lim_{t \to \infty} OBSF_{ki}. \end{aligned} \tag{34}$$

Furthermore, according to Theorem 1, we have the following observation about the effective hit ratios for the SB-PER and the R-CB, respectively:

$$p_{SB-PER}^U = p_{R-CB}^U = \frac{1}{\mu} \sum_{i=1}^{N} \frac{\mu_{ki}^2}{\lambda_i + \mu_{ki}} = \frac{1}{\mu} \lim_{t \to \infty} \sum_{i=1}^{N} OBSF_{ki}. \tag{35}$$

Therefore, the OBS factor actually defines an upper bound of the effective hit ratio of a particular data item.

Theorem 2 suggests that the upper bounds of the effective hit ratios can actually be reached when there is no replacement. The proposed replacement policy "*OBS*" always tries to keep the data items with larger OBS factors in the cache and to evict the data item with the least OBS factor. Therefore, the data items with larger OBS factors tend to stay in the cache longer than those with smaller OBS factors and their effective hit ratios ($p_{SB-PER,i}$ and $p_{R-CB,i}$) approach their upper bounds in Lemma 1. This claim is verified through simulations and shown in Section 7.

The above analysis can also be applied to the communication cost of the SB-PER due to Corollaries 1 and 3. According to Corollaries 1 and 3, we have

$$\begin{aligned} c_{SB-PER}^L &= c_{req} + \left(1 - p_{SB-PER}^U\right)c_{obj} + p_{SB-PER}^U c_{ack} \\ &= c_{req} + c_{obj} - \frac{1}{\mu}\left(c_{obj} - c_{ack}\right) \lim_{t \to \infty} \sum_{i=1}^{N} (OBSF_{ki}). \end{aligned} \tag{37}$$

It implies that the communication cost lower bound of the SB-PER becomes less when data items with larger OBS factors are in the cache. This is also confirmed by Corollary 3. When there is no replacement, the communication cost is exactly the lower bound given in Corollary 1. The OBS always tries to keep data items with large OBS factors in the cache and to evict data items with small OBS factors. The actual communication cost of the data items with large OBS factors tends to be small.

As to the R-CB, we have the following observations according to Corollary 2:

$$\begin{aligned} c_{R-CB}^L &= \left(1 - p_{R-CB}^U\right)\left(c_{req} + c_{obj} + c_{inv} + c_{ack}\right) \\ &= \left(c_{req} + c_{obj} + c_{inv} + c_{ack}\right) \\ &\quad - \frac{1}{\mu}\left(c_{req} + c_{obj} + c_{inv} + c_{ack}\right) \lim_{t \to \infty} \sum_{i=1}^{N} OBSF_{ki}. \end{aligned} \tag{38}$$

Though the lower bound cannot actually be reached, the difference is no more than two message costs. Considering that the data item sizes are large, we can have a similar argument as the SB-PER.

In summary, the OBS replacement policy favors caching the data items with good bounds, that is, the OBS has the capability of keeping good data items and evicting bad data items in the cache. When these data items are cached, the actual effective hit ratios of the data items actually gradually approach the found bounds. The sum of the effective hit ratios of these data items must be larger than those of the data items otherwise chosen. The performance evaluation presented in Section 7 confirms the effectiveness of the OBS replacement policy.

6 Comments on Replacement Policies of Network Data Applications

Cache replacement policies play an important role when the cache size is very limited. Many MTs have small memory when compared with the data items being accessed. Therefore, replacement policies are important aspects of mobile/wireless data caching.

The least recently used (LRU) replacement policy is popular in operating systems and computer architecture areas because data and programs are often stored continuously in memory or hard disks. The access pattern forms the so-called locality. Moreover, the LRU can be regarded as the simplest replacement policy. For data applications where the data access pattern is not bound with their location (in memory or hard disks), the LRU is not necessarily a good choice. As shown in [22], bursty access patterns make the LRU perform inadequately. Furthermore, it could be the worst choice, as shown in [11], for some database systems queries. It will be beneficial to study different replacement policies for network data applications where a busty access pattern is not uncommon. Many frequency-based replacement policies have been designed, studied, and even implemented for real systems, for example, the SQUID Web proxy cache system [2], [10], [12], [22], [23], [26], [29], [30]. An example of such a replacement policy is the LFU. The goal of this paper is to use both update and access frequencies to design a simple and effective replacement policy.

Many cache replacement policy evaluations use network traces, which are representations of real data collected from real networks. However, network traces are only samples and the performance evaluations of different traces can vary from trace to trace, as shown in [5]. In this paper, we use Poisson distributions to model the network data accesses and updates. Poisson distribution may not be a precise model but it is indeed a generalization of network data accesses and updates. It has been used quite often in research of this kind [17], [29], [30].

In our study, we assume that data items are of the same size. First, as observed in [5], the effectiveness of data item size used in replacement policies diminished for some network traces. Second, our paper is aimed at network data applications, where memory pages can be regarded as a unit of data item for a non-Web application. Third, the combinatorial optimization problem of multiple data items of different sizes for the cache replacements is NP-hard [30]. We leave this problem to our future work.

7 Performance Evaluation

In this section, we use discrete-event simulation to evaluate the performance of the proposed replacement policy, the OBS, and also compare its performance with that of the LFU [23] by using perfect access frequency information [26]. We developed our simulation programs in C++.

7.1 Performance Metrics

The design goals of our replacement policy are to increase the effective hit ratio and to reduce communication cost. During the simulations, we collect the following measurements to calculate the effective hit ratios and the communication costs for both the SB-PER and the R-CB with the OBS or the LFU.

Let $n_{a,j}$ and $n_{u,j}$ denote the total number of accesses to data item O_j at the client and the total number of updates to data item O_j at the server, respectively. Let n_{miss} be the total number of cache misses and $n_{inv-hits}$ be the total number of cache hits to invalid data items. Note that, whenever a cache miss or a cache hit to an invalid data item happens, the accessed data item is sent from the server to the client. When the R-CB is exercised, an invalidation message is delivered to clients, where the data item being updated is cached. Let $n_{inv-msg}$ represent the total number of invalidation messages. Let n_{rep} denote the total number of cache replacements. Let n_a denote the total number of accesses of all data items. It is obvious that we have $n_{rep} \leq n_a$ and $n_{miss} \leq n_a$. Similarly to [20], we can compute the effective hit ratios in the simulations as follows:

$$p_{SB-PER} = p_{R-CB} = 1 - \frac{n_{miss} + n_{inv-hits}}{n_a}, \quad \text{where } n_a = \sum_{i=1}^{N} n_{a,j}. \tag{39}$$

We compute the communication cost for the SB-PER, that is, c_{SB-PER}, and that for the R-CB, that is, c_{R-CB}, by using (16) and (21), respectively. Note that, for the R-CB, we also have

$$p_{inv} = \frac{n_{inv-msg}}{n_a}, \tag{40}$$

$$p_{rep} = \frac{n_{rep}}{n_a}. \tag{41}$$

7.2 Simulation Setup

The server has N data items and a client equips a cache which can hold up to K data items. Assume that access and update events are Poisson processes. An access always happens at a client and an update always occurs at the server. The rates of the access and the update to the data item O_j are μ_j and λ_j, respectively. We denote total access rate and total update rate as μ and λ, respectively, that is, $\mu = \sum_{i=1}^{N} \mu_i$, and $\lambda = \sum_{i=1}^{N} \lambda_i$.

Two issues are worth noting:

- N is not necessarily the actual database size because the database may contain data items that are never accessed by a particular client in a certain time frame in which we are interested. Therefore, N is actually an "effective database size." For mobile/wireless clients, N can be relatively small, even though the actual database may be huge. To reflect this fact in our simulation studies, any access rate of the N data items must be larger than 0 and N takes a relatively small value.
- Update frequencies are global statistics measured at the server, whereas access frequencies are local statistics collected for clients. Therefore, the update rate of a data item can be larger than the access rate of a data item because more than one client can have updated the data item between two consecutive accesses to the data item at a client.

It has been observed that, in Web applications, different data items have different popularities. To reflect this observation, we let access events of different data items follow cutoff Zipf-like distributions [4]. It is believed to be a reasonable assumption for Web applications and a representation for disparity of access popularity, though it may not hold for any network data application. We let the update rates of different data items become different. For simplicity, we assume that the update events also follow cutoff Zipf-like distributions. N data items are always ranked as $i = 1, 2, \dots, N$. Thus, when an access or an update happens, the data item O_k whose rank is i is chosen to be accessed or updated at probability $p_i = 1/[i^\alpha \sum_{j=1}^{N} 1/j^\alpha]$.

Note the following remarks:

- We define α ($\alpha \geq 0$) as a Zipf exponent. When $\alpha = 0$, $p_i = 1/N$, that is, all data items are chosen at the same probability $1/N$. The access and update events do not necessarily follow the same cutoff Zipf-like distribution. Let α_a denote the Zipf exponent of access events and α_u be the Zipf exponent of update events.
- Data item identifier (ID) and data item rank are different concepts, that is, data item O_i is not necessarily ranked as i.
- Data items can be ranked independently for access and update events. A data item with rank i for access events is not necessarily the data item with rank i for update events. Thus, a data item with a larger access probability is not necessarily a data item with a larger update probability.

Since there are two different Zipf-like distributions for access and update events, we categorize their relationships into five cases:

- Case 1. $\alpha_a > 0$ and $\alpha_u = 0$, that is, the access frequencies of different data items are different and the update frequencies of all data items are the same.
- Case 2. $\alpha_a = 0$ and $\alpha_u > 0$, that is, the access frequencies of all data items are the same and the update frequencies of different data items are different.
- Case 3. $\alpha_a > 0$, $\alpha_u > 0$, and the access and update rankings to the same data item are the same, that is, a more frequently accessed data item is also a more frequently updated data item. Furthermore, the access frequencies of different data items are different and the update frequencies of different data items are different.
- Case 4. $\alpha_a > 0$, $\alpha_u > 0$, the access rankings of data items are in an ascending order, and the update rankings are in a descending order, that is, a more frequently accessed data item is also a less frequently updated data item. Furthermore, the access frequencies of different data items are different and the update frequencies of different data items are different.
- Case 5. $\alpha_a > 0$, $\alpha_u > 0$, and the access and update rankings of data items are totally independent, that is, if we order the data items according to access probabilities, the update probabilities tend to be randomly distributed. Furthermore, the access frequencies of different data items are different and the update frequencies of different data items are different.

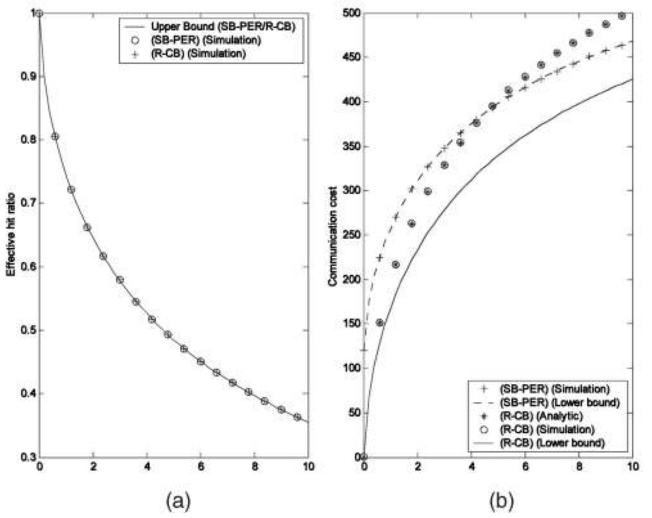

Fig. 3. Comparison of analytical and simulation results for both SB-PER and R-CB. (a) Effective hit ratio versus total update rate (λ). (b) Communication cost versus total update rate (λ).

All of the above cases are studied in the simulations. We let $c_{req} = c_{ack} = c_{inv} = c_a = 60$, which is a reasonable value for many wireless systems such as General Packet Radio Service (GRPS) and iSMS [20]. Since these message sizes are the same, we use $c_{msg} = 60$ to denote the message size. Unless otherwise stated, we choose $c_{obj} = 10c_{msg} = 600$, $N = 20$, $K = 10$, $\mu = 1.0$, $\lambda = 1.4$, $\alpha_a = 0.1$, and $\alpha_u = 1.8$. Though these default parameters are chosen to capture the scenario where data items have different access rates and a small portion of the entire data set has very high update rates, we vary each parameter in Section 7.5 (for example, α_u from 0 to 1 and λ from 1 to 5) and study their effects on the performance. We start the simulation when the client cache is empty. We obtain the metrics measurements when the system has reached a stable state for a long time. To meet this requirement, we often run the simulation for 10^7 arrival events.

7.3 Comparison of Simulation and Analytical Results

We compare the analytical and simulation results for the SB-PER and the R-CB. We choose $K = 20$ and $\alpha_a = 0$. This is Case 2 and there will be no replacement. The effective hit ratios and communication costs are obtained accordingly for both the SB-PER and the R-CB. Fig. 3a shows the effective hit ratio upper bounds and the effective hit ratios from simulations for both the SB-PER and the R-CB. Fig. 3b shows the communication upper bounds and the communication costs from simulations for both the SB-PER and the R-CB.

We have the following observations:

- Fig. 3a shows that the effective hit ratios of the SB-PER and the R-CB are the same in the given case and they match the effective upper bounds defined in Theorem 1. It therefore confirms Theorem 2. Fig. 3b

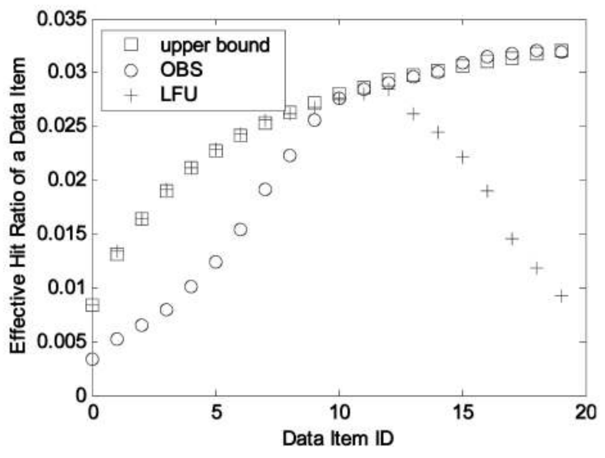

Fig. 4. Effects of on-bound selection.

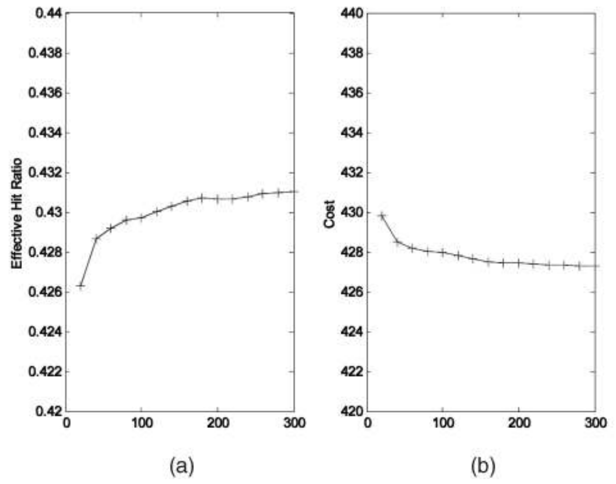

Fig. 5. Performance of SB-PER+OBS versus sliding window size. (a) Effective hit ratio versus sliding window size. (b) Communication cost versus sliding window size.

shows that the communication upper bounds of the SB-PER defined in Corollary 1 match the simulation results obtained in the given case. This confirms Corollary 3. The communication upper bound of the R-CB defined in Corollary 2 is less than both the analytic and simulation results obtained in the given case, but the difference is less than two message costs. This confirms Corollary 4.

- Even though the cache is sufficiently large to hold the entire database, the effective hit ratios decrease, whereas the communication costs increase for both the SB-PER and the R-CB when λ increases, as shown in Figs. 3a and 3b. This is because some cached data items are invalidated by updates. It implies that it is important to consolidate the update process into replacement policies so that frequently updated but infrequently accessed data items will not be cached. Our analytical analysis provides an understanding of what a frequently updated but infrequently accessed data item can be.

7.4 Effects of "OBS"

The simulation results shown in Fig. 4 demonstrate the effects of "*OBS*." As an example, we only have the result obtained for Case 3, where $\alpha_u = 1.0$. In Fig. 4, the data items are assigned ID numbers according to their upper bounds, that is, their OBS factors. For example, the data item with the least ID has the least OBS factor. When the OBS is exercised, K data items with the most upper bounds almost achieve the most, that is, the "upper bound" in the figure. However, when the LFU is exercised, the opposite result is observed. This figure confirms that the OBS indeed keeps a "*good*" data item in the cache.

Interestingly, none of the data items has zero effective hit ratios. This is due to the existence of updates. A cached data item always has a chance of being invalidated and its cache space can be used to cache other data items no matter what replacement policy is used and what its update and access rates are.

7.5 Frequency Measurement

We adopt a sliding window scheme [10], [12], [30] for calculating the access and update frequencies in the algorithms. For each data item, we measure the k most recent access times. We calculate the access frequency of data item i as follows: $f_i^a = \frac{k}{t - t_{i,k}^a}$, where the superscript "a" denotes access, t is the current time, and $t_{i,k}^a$ is the time of the kth most recent access on data item O_i. Similarly, we calculate the update frequency of data item i as follows: $f_i^u = \frac{k}{t - t_{i,k}^u}$, where the superscript "u" denotes update. When $k = 1$, f_i^a is directly related to the access recency of the data item. Therefore, by adjusting k for access to data item O_i, f_i^a can be a measurement for both access frequency and access time (that is, access recency). Similarly, by adjusting k for updates to data item O_i, f_i^u can be a measurement for both update frequency and update time (that is, update recency).

Fig. 5 shows that the defined metrics vary with the sliding window size. All five cases described in Section 7.2 show similar results. For demonstration purposes, we only show the result of Case 3, in which the performance slightly varies with different window sizes and the performance becomes almost invariant with window size when the window size is large enough.

Denote the sliding window size as k. It appears that the overall memory space needed for the above scheme is on the order of $O(kN)$ to store most k recent access (or update) times of N data items. The scheme seems to become prohibitive when k is large. The following example shows that it is not difficult to overcome it: From observation of Fig. 5, when k is large enough, a similar performance can be obtained when we use $k/2$ as the window size. We only keep two access stamps and two access counters for data item O_i. We denote $t^a_{i,n_{a,i}=0 \bmod k}$ as the access timestamp of data item O_i when $n_{a,i} = 0 \bmod k$, where $n_{a,i}$ is the total number of accesses on data item O_i and "mod" is modular operation. Let $t^a_{i,n_{a,i}=0 \bmod k/2}$ be the access timestamp of data item O_i when $n_{a,i} = 0 \bmod k/2$. We reset counter $k_{i,n_{a,i}=0 \bmod k}$ at time $t^a_{i,n_{a,i}=0 \bmod k}$ and counter $k_{i,n_{a,i}=0 \bmod k/2}$ at time $t^a_{i,n_{a,i}=0 \bmod k/2}$ respectively. We then calculate the access frequency as follows: $f_i^a = k_{i,n_{a,i}=0 \bmod k} \big/ t - t^a_{i,n_{a,i}=0 \bmod k}$ when

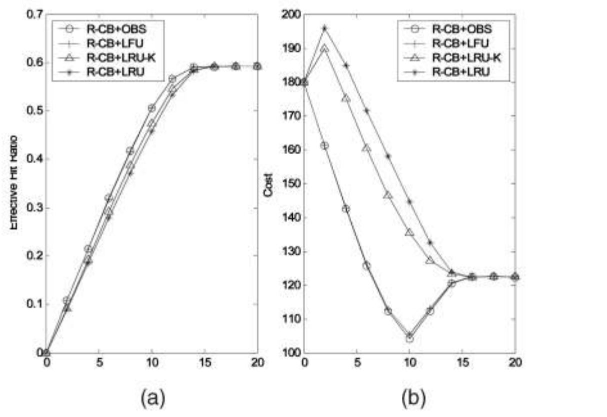

Fig. 6. Performance of R-CB and replacement policies. (a) Effective hit ratio versus cache size (K). (b) Communication cost versus cache size (K).

$k_{i,n_{a,i}=0 \bmod k} \geq k/2$, and $f_i^a = k_{i,n_{a,i}=0 \bmod k/2}/t - t^a_{i,n_{a,i}=0 \bmod k/2}$ when $k_{i,n_{a,i}=0 \bmod k/2} \geq k/2$. Then, the space complexity becomes $O(N)$, which is the same as the LRU.

The above sliding window approach can be regarded as a very simple aging method. Aging [10], [12] is a well-known mechanism for considering both access frequency and the age of a cached data item (access time or recency). A well-designed aging method not only helps reduce the memory space requirement, but also helps cope with the evolving access (update) pattern. Many different aging methods [10], [12] have been proposed. Some cache replacement policies, such as LRU-K [22], have built-in aging mechanisms. In our study, aging has to be considered for both the access and update processes. Better aging methods can help our proposed replacement policy better cope with the evolving access (update) pattern. They deserve dedicated study due to their complexity. However, they are not the focus of this paper and we leave it as our future work.

7.6 Comments on the Importance of Cache Replacement Policies

MTs are gradually gaining more memory which can be used for client caching. We aim at reducing the overall network bandwidth usage. For this purpose, cache replacement policies cannot be ignored, even for MTs with large cache space. This argument holds true for cache mechanisms which involve CB procedures. We refer to the CB procedure as the procedure in which the server informs clients that cached data items have become invalid due to updates. The CB (for example, R-CB in this paper), Lease, and IR schemes use the CB procedure. Our argument is verified in Fig. 6. We choose $c_{req} = c_{ack} = c_{inv} = c_a = 60$, $c_{obj} = 2c_{msg} = 120$, $N = 20$, $K = 10$, $\mu = 1.0$, $\lambda = 5$. These 20 data items are divided into four groups. The access rates (update rates) of the data items within the same group are the same. The access rates and updates are drawn from two different Zip-like distributions, with $\alpha_a = 0.1$, and $\alpha_u = 10$; however, the rankings of the four groups for access and update processes are of opposite directions (similar to Case 4 described in Section 7.2). This parameter setting ensures that some data items are moderately frequently accessed but are updated very frequently and the cost of sending invalidation messages is quite large. Fig. 6 compares four cache replacement policies, that is, OBS, LFU, LRU, and LRU-K [22], where K is taken to be 2 when R-CB is exercised. The following observations are interesting:

- Fig. 6a shows that the effective hit ratios of all four replacement policies increase with the cache size.
- Fig. 6b shows that the minimum communication cost is obtained when $\mathrm{K} = 10$ for R-CB+OBS and R-CB+LFU when the cache size is varying. The server needs to send an invalidation message when an update happens to it. It is likely that the client has to fetch a data item from the server due to frequent updates. Then, the cost of sending an invalidation message may not be offset by using the cached copy. In our parameter setup, when the cache space is larger than 10 data items, such types of frequently updated data items will inevitably be cached. The total communication cost goes up while more such data items are cached. Fig. 6b also shows that R-CB+LRU and R-CB+LRU-K have larger costs when given a small cache space. This can be explained similarly.
- Figs. 6a and 6b also show that R-CB+LFU/OBS has better performance than R-CB+LRU/LRU-K. This is due to 1) the nonexistence of the locality related to the adjacency of memory location where the data items are stored (in other words, recency does not capture the data access pattern well) and 2) they do not consider updates. This has been discussed in Section 6. Due to page limits, we will not further compare our replacement policy with LRU and LRU-K in this paper.

In a nutshell, cache replacement policies are not necessarily good for the overall network bandwidth usage when maximizing cache space. Nevertheless, cache replacement policies are essential to a network environment with the existence of updates, even when memory capacity is abundant.

7.7 Case Studies: Comparison of OBS and LFU

The following simulations will show that the OBS outperforms the LFU in terms of both effective hit ratio and communication cost in all five cases listed in the previous subsection. This aspect is shown with regard to different parameters, that is, the Zipf-exponents, cache and database sizes, and access and update rates.

7.7.1 Zipf Exponents

The two schemes are compared when the Zipf exponents are varying. The results are shown in Fig. 7.

- Figs. 7a and 7b show an example of Case 1, that is, $\alpha_a > 0$ and $\alpha_u = 0$. In Figs. 7a and 7b, (SB-PER)/(R-CB)+OBS have the same effective hit ratios and costs as (SB-PER)/(R-CB)+LFU, respectively. It means that OBS and LFU perform the same in this example. In fact, according to (1), the OBS is equivalent to the LFU if the update frequencies of data items are the same, which has been confirmed by this example, since the update frequencies of data items are indeed the same in this case.
- Figs. 7c and 7d show an example of Case 2, that is, $\alpha_a = 0$ and $\alpha_u > 0$. They show that (SB-PER)/(R-CB)+OBS have consistently larger effective hit ratios and consistently smaller costs than (SB-PER)/(R-CB)+LFU, respectively. In other words, the OBS outperforms the LFU.

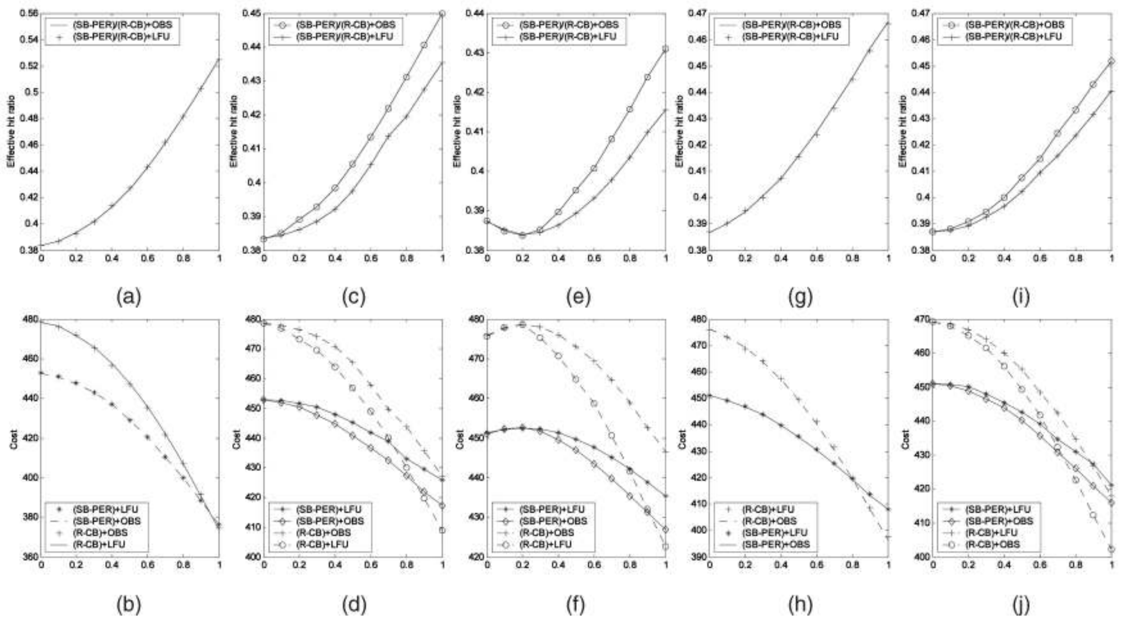

Fig. 7. Performance of the OBS and the LFU at different cases with regard to the Zipf exponents. (a), (c), (e), (g), and (i) show effective hit ratio versus Zipf exponent of access events (α_a) for Cases 1, 2, 3, 4, and 5, respectively. (b) , (d), (f), (h), and (j) show communication cost versus Zipf exponent of access events (α_a) for Cases 1, 2, 3, 4, and 5, respectively.

- Figs. 7e and 7f show an example of Case 3, that is, $\alpha_a > 0$, $\alpha_u > 0$, and the ranking of access and update to the same data item are the same. We observe similar results as in Case 2, that is, (SB-PER)/(R-CB)+OBS have consistently larger effective hit ratios and less costs than (SB-PER)/(R-CB)+LFU, respectively. It means that the OBS chooses appropriate data items to evict from the cache so that better performance is achieved, even if the frequently accessed data items are also frequently updated data items.
- Figs. 7g and 7h show an example of Case 4, that is, $\alpha_a > 0$ and $\alpha_u > 0$ in which the rankings of data items for accesses are in ascending order and those for updates are in descending order. For example, frequently accessed data items are also infrequently updated data items. The OBS tends to cache frequently accessed data items but not infrequently updated data items, which is consistent with the LFU in this particular case. Therefore, it is not surprising that the OBS and the LFU perform the same.
- Figs. 7i and 7j show an example of Case 5, that is, $\alpha_a > 0$ and $\alpha_u > 0$, in which the rankings of data items for both accesses and updates are totally independent. Because access and update probabilities are assigned randomly, every different run possibly produces a different result. Thus, the average performance should be more representative than an individual run. Figs. 5i and 5j show the average performance out of 50 different runs. It is clear that the (SB-PER)/(R-CB)+OBS have larger effective hit ratios and less communication costs than (SB-PER)/(R-CB)+LFU, respectively.

All five of these possible cases show the advantage of the OBS over the LFU, which confirms that the OBS, indeed, is capable of choosing "the bad data item" to be evicted while retaining "the good data item," regardless of the Zipf exponents.

We also have the following observations:

- As shown in Figs. 7a, 7c, 7g, and 7i, when α_u increases, the effective hit ratio increases as well. α_u's increasing means that fewer data items take up most of the total update rate and others have smaller update rates, even though the total update rate is the same. The OBS tends to cache data items with smaller update rates. Those few data items with larger update rates are probably cached a smaller number of times. The most cached data items have smaller update rates. Therefore, the cached data items are less likely to be made invalid by updates and the effective hit ratio is increased.
- As shown in Figs. 7b, 7d, 7h, and 7j, when α_u increases, the communication cost decreases. It is due to the same reason as the above.
- Fig. 7e shows that the effective hit ratio decreases first and then increases when α_u increases. Fig. 7f shows that the communication cost increases first and then decreases when α_u increases. Basically, the increase of the effective hit ratio and the decrease of the communication cost are caused by the reason above.

7.7.2 Cache and Database Sizes

Fig. 8 shows the comparison between the OBS and the LFU when we vary the cache size and fix the database size. As before, five different cases are studied. We have the following observations:

- Figs. 8a and 8b show an example of Case 1, that is, $\alpha_a > 0$ and $\alpha_u = 0$. Both replacement policies have the same effective hit ratios and costs, that is, OBS and LFU perform the same in this example, regardless of the cache size, due to the same reason presented for Figs. 7a and 7b.

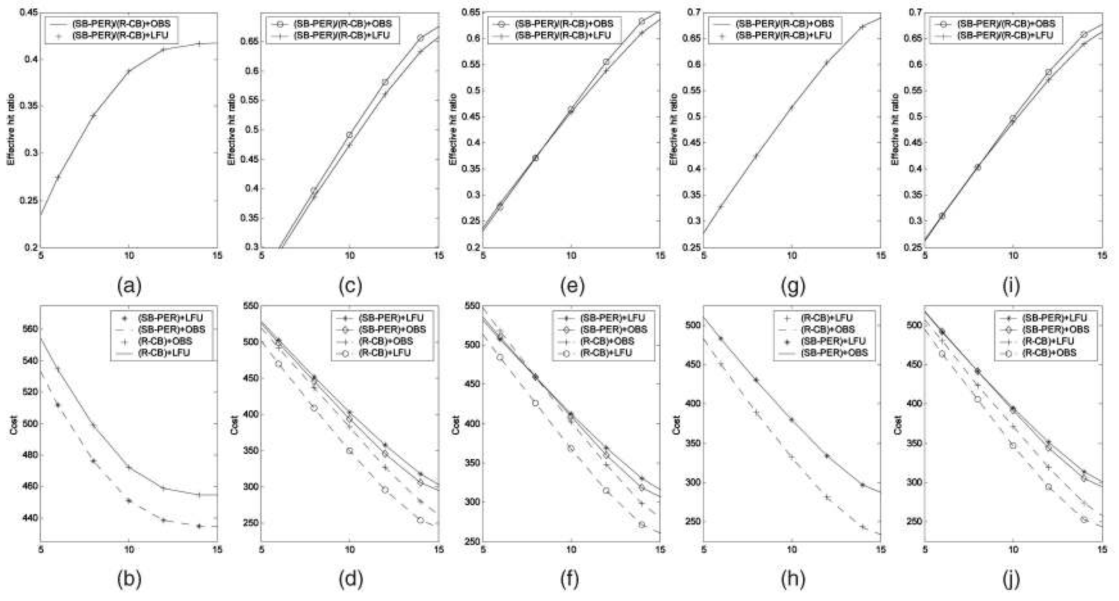

Fig. 8. Performance of the OBS and LFU at different cases with regard to the cache size. (a), (c), (e), (g), and (i) show effective hit ratio versus cache size (K) for Cases 1, 2, 3, 4, and 5, respectively. (b), (d), (f), (h), and (j) show communication cost versus cache size (K) for Cases 1, 2, 3, 4, and 5, respectively.

- Figs. 8c and 8d show an example of Case 2, that is, $\alpha_a = 0$ and $\alpha_u > 0$. OBS has better performance than LFU, regardless of the cache size.
- Figs. 8e and 8f show an example of Case 3, that is, $\alpha_a > 0$, $\alpha_u > 0$, and the ranking of access and update to the same data item are the same. OBS has better performance than LFU, regardless of the cache size.
- Figs. 8g and 8h show an example of Case 4, that is, $\alpha_a > 0.1, 1 > 0$, and $\alpha_u > 0$, in which the rankings of data items for accesses are in ascending order and those for updates are in descending order. For example, frequently accessed data items are also infrequently updated data items. It is not surprising that the OBS and the LFU perform the same.
- Figs. 8i and 8j show an example of Case 5, that is, $\alpha_a > 0$ and $\alpha_u > 0$, in which the ranking of data items for both accesses and updates are totally independent. The results are the average performance out of 50 different runs. It is clear that the (SB-PER)/(R-CB)+OBS have larger effective hit ratios and less communication costs than (SB-PER)/(R-CB)+LFU, respectively.

Besides the above observations, it is clear that, from the simulation results shown in Fig. 8, the effective hit ratios of (SB-PER)/(R-CB)+OBS/LFU increase with the cache size. In our examples, the communication costs decrease when the cache size increases. This is because, when the cache size increases, more data items are cached. However, no matter how the cache size varies, we have shown that (SB-PER)/(R-CB)+OBS have better performance than (SB-PER)/(R-CB)+LFU.

We study how performances of (SB-PER)/(R-CB)+OBS/LFU vary with the database size when the cache size is fixed. The results are shown in Fig. 9, where $K = 10$:

- Figs. 9a and 9b show an example of Case 1 and Figs. 9g and 9h show an example of Case 4. These two examples show that the OBS and the LFU have the same performance, whether the SB-PER or the R-CB is exercised. It is due to an argument similar to the one shown in Figs. 7a, 7b, 7g, and 7h.
- Figs. 9c and 9d, 9e and 9f, and 9i and 9j show examples for Cases 2, 3, and 5, respectively. Evidently, the OBS consistently outperforms the LFU, whether the SB-PER or the R-CB is exercised.

These observations evidently show that the OBS is a better replacement policy than the LFU in terms of effective hit ratio and communication cost.

Fig. 10 shows the performance varying with N when K/N is a constant. For demonstration purposes, only Case 3, where $\alpha_u = 1.0$, is shown. Evidently, Figs. 10a and 10b show that OBS always performs better when the database and cache sizes scale up.

7.7.3 Access and Update Rates

In this part, we study the effects of access and update rates, that is, μ and λ. Fig. 11 shows the simulation results, where μ is fixed and λ is varying. We can observe similar results as before:

- Figs. 11a and 11b and 11g, and 11h correspond to Cases 1 and 4, respectively. They show that the OBS and the LFU have almost the same performance, regardless of λ, when either the SB-PER or the R-CB is exercised. Figs. 11c and 11d, 11e and 11f, and 11i and 11j show the results for Cases 2, 3, and 4, respectively. They show that the OBS outperforms the LFU, regardless of λ, when either the SB-PER or the R-CB is exercised.

The above two observations show that the OBS is either better than or equivalent to the LFU, regardless of λ. Besides, no matter the case, the effective hit ratios of (SB-PER)/(R-CB)+OBS/LFU decrease and the communication costs of (SB-PER)/(R-CB)+OBS/LFU increase when λ increases. When λ increases, a data item has a greater chance of being updated. If the data item is cached, it has a greater chance of being invalidated. Therefore, the effective

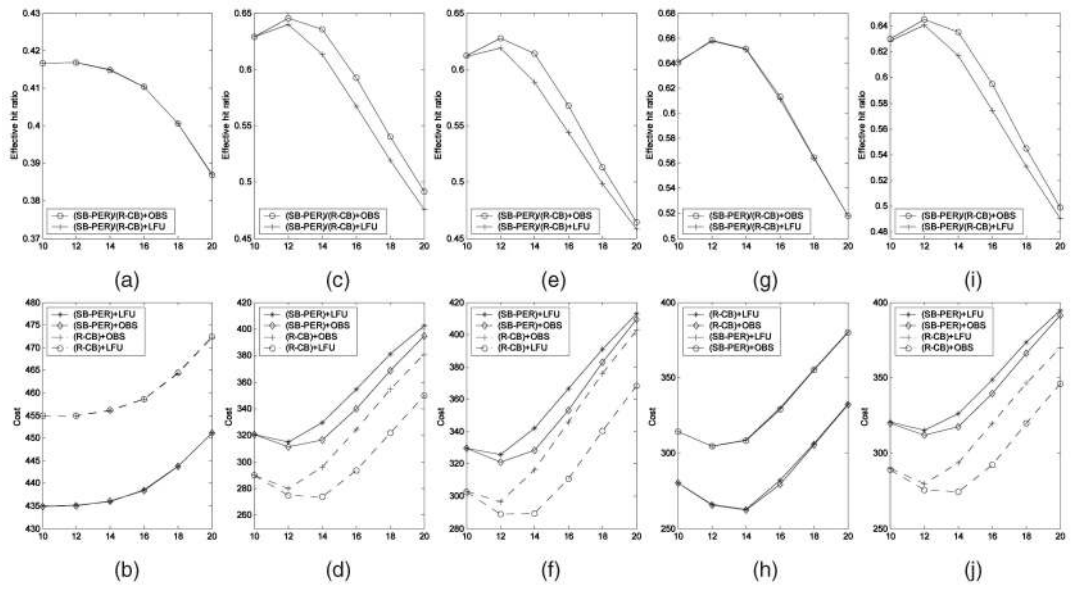

Fig. 9. Performance of the OBS and LFU at different cases with regard to the database size when the cache size is fixed. (a), (c), (e), (g), and (i) show effective hit ratio versus database size (N) for Cases 1, 2, 3, 4, and 5, respectively. (b), (d), (f), (h), and (j) show communication costs versus database size (N) for Cases 1, 2, 3, 4, and 5, respectively.

hit ratios of (SB-PER)/(R-CB)+OBS/LFU become less and the communication costs of those become more.

7.8 Comments on the Server-Based Approach

In order to reduce transmission cost over wireless links, we have introduced the server-based cache access algorithm, that is, the SB-PER, to test our update-based replacement policy, that is, the OBS. The SB-PER is capable of providing up-to-date access and updated information to the replacement policy. In general, the SB-PER requires more powerful servers. Since servers are much more inexpensive compared to wireless links, we assume that a server can be easily upgraded with a more powerful machine or a cluster of machines, that is, a virtual server or a server farm. Furthermore, when compared with the client-based cache access algorithm, the SB-PER improves the effective hit ratio significantly, that is, fewer data items are fetched from the server. Therefore, our algorithm may perform even better in terms of response time.

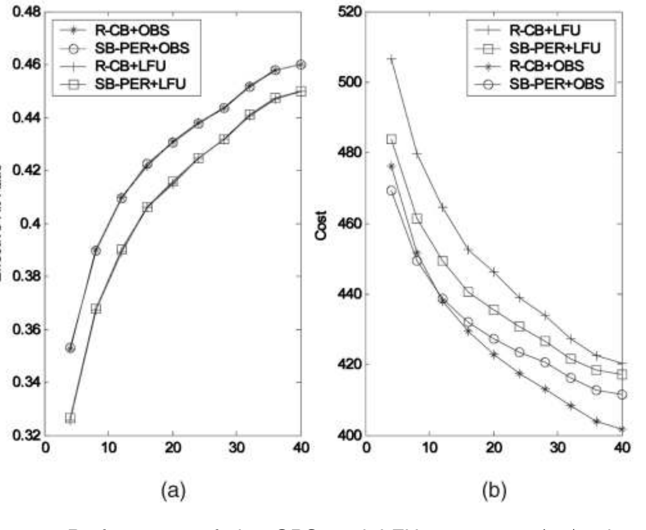

Fig. 10. Performance of the OBS and LFU versus N (K/N is a constant). (a) Effective hit ratio versus sdatabase size (N). (b) Communication cost versus database size (N).

Server response time is related to many factors. CPU cycles are on the order of microseconds (μs) or even less, and the hard disk random-seek time is on the order of a few milliseconds (ms). Therefore, the hard disk random-seek time is often a dominant factor for server response times [25]. We assume that the server stores object profiles and update profiles in the main memory; therefore, accesses to two of these profiles do not result in a random seek on hard disks. As shown in Fig. 2a, a random seek is performed to locate the data item on the hard disk only when the data item is cached or the data item is invalid (Steps 2.4 and 3.2). For a fair comparison, we assume that metadata including timestamps of data items can be loaded into the main memory as well when the PER is exercised so that the PER requires a random seek when the data item is not cached or invalid too. Fig. 12 compares the random-seek count per access between the PER and the SB-PER (only Case 5 is shown). However, the PER is exercised with the LFU and the SB-PER uses the OBS. A different replacement policy is chosen for the PER because the PER does not provide update information to the clients and the clients cannot use the OBS. Fig. 12 shows that the SB-PER has a much smaller random-seek count per access. In other words, the SB-PER outperforms the PER in terms of server response time under our parameter settings.

7.9 More General Network Traffic

In our analytical analysis in Section 5, data access and update follow a Poisson distribution. In this section, we loosen the assumption of Poisson distribution to demonstrate the performance of the proposed OBS under a more general network traffic model. Fig. 13 is obtained when both access and update processes follow Gamma distributions. Although similar results can be obtained for other cases, the figure only shows Case 3, as defined in Section 7.2. In Figs. 13a and 13b, we vary the variance of access time. The results show that 1) the effective hit ratio and the

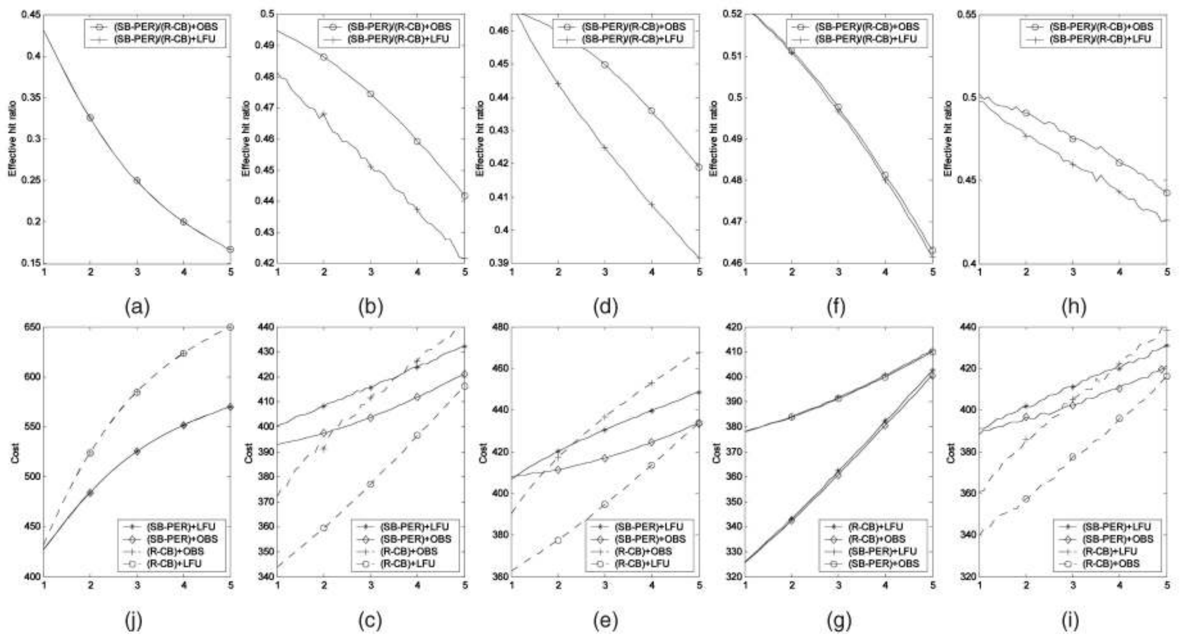

Fig. 11. Performance of the OBS and LFU at different cases with regard to the total update rate when the total access rate is fixed. (a), (c), (e), (g), and (i) show effective hit ratio versus total update rate (λ) for Cases 1, 2, 3, 4, and 5, respectively. (b), (d), (f), (h), and (j) show communication costs versus total update rate (λ) for Cases 1, 2, 3, 4, and 5, respectively.

transmission costs of SB-PER/R-CB+OBS/LFU do not vary significantly with the variance and 2) the OBS replacement policy consistently outperforms the LFU, regardless of whether the SB-PER or the R-CB is exercised when the variance of access time is varying. This suggests that the proposed OBS can be applied to more general network traffic and consistently give better performance even though it is based on a more stringent assumption.

8 Conclusion

An update can make a cached data item obsolete and, thus, a cache hit becomes useless. Little research has consolidated the update process into a cache algorithm design, especially the cache replacement policy design. In this paper, we proposed a replacement policy, the OBS, which uses both access and update frequencies. We conducted an analytical analysis of cache access and update. An upper bound of the effective ratio and a lower bound of communication cost are given. We further show that the upper bound of the effective hit ratio and the lower bound of the communication cost can be reached when there is no replacement. According to our understanding of these bounds, we show that the proposed replacement policy tries to keep the good data items in the cache and to evict the bad data items from the cache in terms of increasing the effective hit ratio and reducing the communication cost. We studied its performance with two cache access algorithms, the SB-PER and the R-CB, through discrete-event simulations. The simulation results show that the OBS outperforms the LFU in terms of both the effective hit ratio and the communication cost in all of the possible cases that we have studied. Therefore, we believe that the OBS indeed asymptotically keeps the good data items in the cache while evicting the bad ones.

In the simulations, we further loosen the traffic assumption with Gamma distributions and similar results are observed. In our future work, we will address the effects of different data item sizes and extend update-based replacement policies to client-based cache access algorithms.

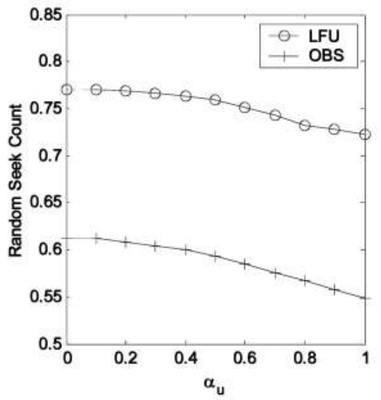

Fig. 12. Random-seek count per access versus α_u.

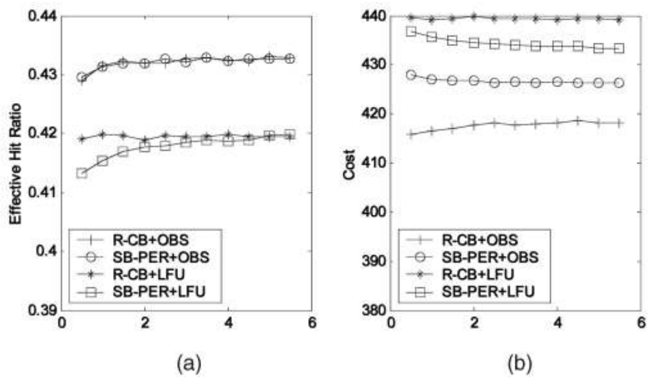

Fig. 13. Metrics of network traffic of Gamma distribution. (a) Effective hit ratio versus the variance of acess time. (b) Communication cost versus the variance of access time.

References

[1] S. Acharya and S. Muthukrishnan, "Scheduling On-Demand Broadcasts: New Metrics and Algorithms," *Proc. ACM MobiCom '98,* pp. 43-54, 1998.

[2] A. Balamash and M. Krunz, "An Overview of Web Caching Replacement Algorithms," *IEEE Comm. Surveys and Tutorials,* vol. 6, no. 2, pp. 44-56, 2004.

[3] D. Barbará and T. Imielińksi, "Sleepers and Workaholics: Caching Strategies for Mobile Environments (Extended Version)," *VLDB J.,* vol. 4, no. 4, pp. 567-602, 1995.

[4] L. Berslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM '99,* vol. 1, pp. 126-134, Mar. 1999.

[5] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications," *World Wide Web J.,* vol. 2, pp. 15-28, 1999.

[6] J. Cai and K.-L. Tan, "Energy-Efficient Selective Cache Invalidation," *Wireless Networks,* vol. 5, no. 6, pp. 489-502, 1999.

[7] G. Cao, "Proactive Power-Aware Cache Management for Mobile Computing Systems," *IEEE Trans. Computers,* vol. 51, no. 6, pp. 608-621, June 2002.

[8] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *IEEE Trans. Knowledge and Data Eng.,* vol. 15, no. 5, Sept./Oct. 2003.

[9] B.Y.L. Chan, A. Si, and H.V. Leong, "Cache Management for Mobile Databases: Design and Evaluation," *Proc. 14th Int'l Conf. Data Eng. (ICDE '98),* pp. 54-63, Feb. 1998.

[10] L. Cherkasova and G. Ciardo, "Role of Aging, Frequency, and Size in Web Cache Repalcement Policies," *Lecture Notes in Computer Science,* vol. 2110, pp. 114-123, 2001.

[11] H. Chou and D. DeWitt, "An Evaluation of Buffer Management Strategies for Relational Database Systems," *Proc. 11th Int'l Conf. Very Large Data Bases (VLDB '85),* pp. 141-172, 1985.

[12] J. Dilley, M. Arlitt, and S. Perret, "Enhancement and Validation of the Squid Cache Replacement Policy," *Proc. Fourth Int'l Web Caching Workshop (WCW '99),* 1999.

[13] C.C.F. Fong, J.C.S. Liu, and M.H. Wong, "Quantifying Complexity and Performance Gains of Distributed Caching in a Wireless Network Environment," *Proc. 13th Int'l Conf. Data Eng. (ICDE '97),* pp. 104-113, Oct. 1997.

[14] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West, "Scale and Performance in a Distributed File System," *ACM Trans. Computer Systems,* vol. 6, no. 1, pp. 51-58, Feb. 1988.

[15] Q.L. Hu and D.L. Lee, "Cache Algorithms Based on Adaptive Invalidation Reports for Mobile Environments," *Cluster Computing,* vol. 1, no. 1, pp. 39-48, Feb. 1998.

[16] A. Iyengar, E. Nahum, A. Shaikh, and R. Tewari, "Web Caching, Consistency and Content Distribution," *The Practical Handbook of Internet Computing,* M.P. Singh, ed., Chapman & Hall/CRC Press, 2005.

[17] R. Jain, *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, 1991.

[18] J. Jing, A.K. Elmagarmid, A. Helal, and R. Alonso, "Bit-Sequences: A New Cache Invalidation Method in Mobile Environments," *Mobile Networks and Applications,* vol. 2, no. 2, pp. 115-127, 1997.

[19] A. Kahol, S. Khurana, S.K.S. Gupta, and P.K. Srimani, "A Strategy to Manage Cache Consistency in a Distributed Mobile Wireless Environment," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 7, pp. 686-700, July 2001.

[20] Y.-B. Lin, W.-R. Lai, and J.-J. Chen, "Effects of Cache Mechanism on Wireless Data Access," *IEEE Trans. Wireless Comm.,* vol. 2, no. 6, pp. 1240-1246, 2003.

[21] M. Nelson, B. Welch, and J. Ousterhout, "Caching in the Sprite Network File System," *ACM Trans. Computer Systems,* vol. 6, no. 1, pp. 134-154, Feb. 1988.

[22] E. O'Neil, P. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 297-306, 1993.

[23] J.T. Robinson and M.V. Devarakonda, "Data Cache Management Using Frequency-Based Replacement," *Proc. ACM SIGMETRICS Conf.,* pp. 134-142, 1990.

[24] K.L. Tan, J. Cai, and B.C. Ooi, "An Evaluation of Cache Invalidation Strategies in Wireless Environments," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 8, pp. 789-807, Aug. 2001.

[25] A.S. Tanenbaum, *Computer Networks,* fourth ed. Prentice Hall PTR, 2003.

[26] J. Wang, "A Survey of Web Caching Schemes for the Internet," *ACM SIGCOMM Computer Comm. Rev.,* vol. 29, no. 5, pp. 36-46, 1999.

[27] K.-L. Wu, P.S. Yu, and M.-S. Chen, "Energy-Efficient Caching for Wireless Mobile Computing," *Proc. 12th Int'l Conf. Data Eng. (ICDE '96),* pp. 336-343, Feb. 1996.

[28] Y. Xiao, "Optimal Location Management for Two-Tier PCS Networks," *Computer Comm.,* vol. 26, no. 10, pp. 1047-1055, June 2003.

[29] J. Xu, Q. Hu, D.L. Lee, and W.-C. Lee, "SAIU: An Efficient Cache Replacement Policy for Wireless On-Demand Broadcasts," *Proc. Ninth ACM Int'l Conf. Information and Knowledge Management (CIKM '00),* pp. 46-53, Nov. 2000.

[30] J. Xu, Q. Hu, W.-C. Lee, and D.L. Lee, "Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 1, pp. 125-139, Jan. 2004.

[31] J. Yin, L. Alvisi, M. Dahlin, and C. Lin, "Volume Leases for Consistency in Large-Scale Systems," *IEEE Trans. Knowledge and Data Eng.,* vol. 11, no. 4, pp. 563-576, July/Sept. 1999.

[32] J. Yuen, E. Chan, K.Y. Lam, and H.W. Leung, "Cache Invalidation Scheme for Mobile Computing Systems with Real-Time Data," *ACM SIGMOD Record,* vol. 29, no. 4, pp. 34-39, 2000.

Hui Chen received the MS and PhD degrees in computer science from the University of Memphis, Tennessee, in 2003 and 2006, respectively. He was a geophysicist and worked as a software developer for AutoZone, Inc., from 2005 to 2007. He joined the faculty of the Department of Mathematics and Computer Science at Virginia State University in 2007. Although he retains his interests in studying computational problems in Earth sciences, he primarily works on computer system and networking research such as the design and analysis of personal communication service systems, wireless LANs, wireless sensors, mobile/wireless distributed systems, and cache systems for wireless systems. He is serving as a guest editor for the *EURASIP Journal on Wireless Communications and Networking* special issue on wireless telemedicine and applications. He is the author of more than 30 scientific papers and articles. He is a member of the IEEE.

Yang Xiao was with Micro Linear as a Medium Access Control (MAC) architect involving the IEEE 802.11 standard enhancement work before he joined the Department of Computer Science at the University of Memphis, Tennessee, in 2002. From 2001 to 2004, he was a voting member of the IEEE 802.11 Working Group. He is currently with the Department of Computer Science at the University of Alabama. He is currently the editor-in-chief of the *International Journal of Security and Networks (IJSN)*, the *International Journal of Sensor Networks (IJSNet)*, and the *International Journal of Telemedicine and Applications (IJTA)*. He is an associate editor or on the editorial boards of several journals, such as the *IEEE Transactions on Vehicular Technology*. He is also a referee/reviewer for many funding agencies, a panelist of the US National Science Foundation (NSF), and a member of the Telecommunications Expert Committee, Canada Foundation for Innovation (CFI). He is a member of the technical program committees of more than 90 conferences, such as IEEE INFOCOM, ICDCS, ICC, Globecom, and WCNC. His research interests are security, telemedicine, sensor networks, and wireless networks. He has published more than 200 papers in major journals (more than 50 in various IEEE journals/magazines), refereed conference proceedings, and book chapters related to his research areas. His research has been supported by the NSF. He is a senior member of the IEEE and a member of the American Telemedicine Association.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.