Optimal Callback with Two-Level Adaptation for Wireless Data Access

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

Abstract—Strongly consistent callback cache mechanisms have been studied for data access in wireless networks. In cache access mechanisms, update information is extremely important since an updated data object in a remote server makes the corresponding data objects invalidated in mobile terminals (MTs), and the data object cache hit information in those MTs becomes almost useless. In this paper, we propose an adaptive access mechanism, called *optimal callback with two-level adaptation.* In the first-level adaptation, cache size in an MT is adaptively adjusted based on *update-to-access-ratio* (UAR), defined as the average number of updates per data object access. The range of the cache size is [0, M], where M is the maximum physical cache size of the MT. Two extreme cases are given as follows: 1) When the UAR is very large so that objects in the cache are always obsolete, the cache should not be used and, therefore, the cache size should be set to zero; 2) when the UAR is zero so that every object in the cache is valid, the cache size should be set to M. Under other situations, the cache size is dynamically changed between 0 and M. Define U-threshold of the UAR for any object, a particular important threshold, as a UAR value, beyond which the object should be not cached at all. The idea of the second-level adaptation is that if an object size is small, sending back the object may be a better choice than sending back an invalidation message when the object is updated. Therefore, when an object is updated at the server, it is sent directly to MTs if the object size is smaller than a threshold, called Push Threshold (T); otherwise, an invalidation message is sent to the MTs. We analytically model cost function for the proposed adaptive scheme as the total traffic involved between the server and an MT per data object access, and the optimal cache size and the optimal T value are obtained simultaneously to minimize the cost function. Furthermore, U-threshold is derived analytically. Both simulations and analytical results are used to study and compare the performance of the proposed scheme with several others under many different scenarios.

Index Terms—Adaptive, cache, callback, strong consistency, wireless data access.

1 Introduction

Technology advances increase popularity and availability of modern mobile wireless networks. Accessing information from mobile handheld devices is demanded by the huge population of users and has been made possible by the computational and visual capability of cost efficient mobile handheld devices and the wide coverage of wireless networks, especially personal communication networks. Services provided through mobile wireless networks are rich, for example, instant content access, banking, e-commerce, and gaming, but suffer from their inherent constraints, such as bandwidth limitation, high cost of radio frequency, and the limited battery power of mobile terminals. High communication cost and latency caused by mobile wireless networks are not relieved due to the increasing number of users and their accesses to multimedia.

Wireless Application Protocol (WAP) and iSMS have been developed to support wireless Internet applications [1], [2], [3], [4], [5], [18], [19]. For example, a wireless terminal can obtain data objects from an application server via the wireless application gateway, which internetworks the wireless network with the IP network [5]. A mobile user can access Internet Web applications through a client/server model. Cache can also be used in the client mobile terminal to buffer frequently/recently used data objects sent from the server [6]. For some applications, a strongly consistent data access protocol must be exercised between the client and the server [5], [8]. In [5], Lin et al. implement a business card service that is a generalization of the phone book feature in mobile devices, and is one of the most popular features in mobile devices.

Cao [9], [10] proposes an *Invalidation Report* (IR) based cache invalidation algorithm, which efficiently utilizes the broadcast bandwidth to intelligently broadcast the data requested by clients. In [5], Lin et al. study two strongly consistent algorithms for wireless data access: *poll-each-read (PER)* and *callback (CB)*, using both analytical models and simulations. In the PER mechanism [11], [12], [13], if the data object that a client requests is not in the cache, the client requests it from the server; otherwise, the client always asks the server to check its validity. If it is valid, the server informs the client and the client uses the data object; if it is not valid, the server sends the updated data object to the client to replace the out-of-date data object in the client cache. In the CB mechanism [10], [11], [12], when a data object is to be changed at a server, the server informs the client, the client marks the data object in the cache as invalid, and the space can be reclaimed to accommodate other data objects.

In this paper, we have three observations for the CB scheme: 1) When the update-to-access-ratio (UAR) is so large that objects in the cache are always obsolete, such as stock information, the cache should not be used and, therefore, the cache size should be set to zero, 2) when the

- *Yang Xiao is with the Department of Computer Science, The University of Alabama, 101 Houser Hall, Box 870290, Tuscaloosa, AL 35487-0290. E-mail: yangxiao@ieee.org.*
- *Hui Chen is with the Department of Computer Science, The University of Memphis, Memphis, TN 38152. E-mail: huichen@ieee.org.*

Manuscript received 21 July 2004; revised 17 Jan. 2005; accepted 5 Apr. 2005; published online 15 June. 2006.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0231-0704.

UAR is zero so that every object in the cache is always valid, the cache size should be set to the maximum physical cache size of the mobile terminal, and 3) when a data object is updated at the server and its size is very small, the data object instead of an invalidation message should be sent back to the client. Therefore, we propose an adaptive access mechanism, called *optimal callback with two-level adaptation*, as described in the abstract. Note that the proposed callback with a two-level adaptation scheme is also called adaptive CB in this paper. We provide an analytical model and study the optimality of the cache size and the Push Threshold value simultaneously, and U-threshold (described in the abstract) is obtained analytically. Simulations are conducted not only to validate the analytical models, but also to provide performance studies under more general assumptions than those in analytical models.

The rest of this paper is organized as follows: Section 2 proposes adaptive data access schemes. Section 3 provides analytical models. Section 4 studies optimality issues. Section 5 provides performance evaluation via both analytical results and simulation results. Finally, we conclude this paper in Section 6.

2 Cache Access Mechanisms

A caching mechanism involves many aspects, among which two important aspects are a cache access scheme and a replacement policy. A cache access scheme defines how a client and a server exchange messages and objects. Examples of cache access schemes are callback, PER, IR, etc. A replacement policy is associated with a cache access scheme to define how to evict an object in the cache when the cache is full. A replacement policy may or may not be related to a cache access scheme. Examples of cache replacement policies are Least Recently Used (LRU), Least Frequently Used (LFU), and many more. The proposed callback with two-level adaptation is an access scheme, but not a replacement policy. In other words, any replacement policy can be applied to the proposed access scheme. However, in our analytical model introduced in the next section, we adopt the LRU replacement policy as an example. But, our proposed callback with two-level adaptation scheme is not limited to LRU.

How the objects are ranked after being visited totally depends on the replacement policy. For example, the LRU policy ranks visited objects based on the recent time being visited, and the LFU policy ranks visited objects based on the frequency of visited objects during a time period. The rank is for defining how to replace or evict an object from the cache when the cache is full. The number of objects in the server is always larger than (or equal to) the number of objects in the cache.

Next, we describe the proposed adaptive data access scheme and several related schemes.

2.1 Push, Invalidation, and Fetch

There are several callback (CB) schemes. The scheme described in the Introduction as well as in [5] is referred to as the *Invalidation* scheme in this paper. We define another callback scheme, called the *Push* scheme, in which, when a data object is updated at the server, the object is sent to the client instead of an invalidation message. The Invalidation scheme and the Push scheme are two special cases of the callback scheme.

We further define another scheme, called the *Fetch* scheme, in which the cache is disabled. Specifically, when a client needs a data object, it always requests the data object from the server that sends the data object to the client; when an object is updated at the server, no further procedure is needed.

2.2 Optimal Callback with Two-Level Adaptation

In the proposed optimal callback scheme, we consider both cache size and object size. Assume that the maximum physical cache size in a mobile terminal is M in terms of the number of objects it can hold. The range of the cache size is [0, M]. In the proposed scheme, the cache size is adaptively adjusted based on update-to-access-ratio (UAR), defined as the average number of updates per data object access. One extreme case is that when the UAR is so large that objects in the cache are always obsolete, the cache should not be used and, therefore, the cache size should be set to zero. In other words, the Fetch scheme should be used. Note that the Fetch scheme can be treated as a special case when the cache size is zero. Another extreme case is that when the UAR is zero, the cache size should be M. Under other situations, the cache size is dynamically changed between 0 and M. Define the U-threshold of the UAR for any object, a particularly important threshold, as a UAR value, beyond which the object should be not cached at all. If all objects' UARs are beyond the U-threshold, the cache should be disabled, i.e., the optimal cache size is zero. Note that if all objects' UARs are beyond the U-threshold, objects in the cache are not necessarily obsolete. In other words, the U-threshold further strengthens the first extreme case mentioned above. When the UARs of some objects are smaller than the U-threshold and some of them are not, the cache size is dynamically changed between 0 and M, and those objects whose UARs are smaller than the U-threshold are candidates for caching.

Let $O_j, (j = 1, \ldots, N)$ denote the data object j, where N is the number of data objects. Let $U_j, (j = 1, \ldots, N)$ denote the UAR value of the object j. We have $U_j = \lambda_j/\mu_j, (j = 1, \ldots, N)$, where λ_j and μ_j are the update rate and access rate of the object j, respectively. The UAR is defined as $U = \lambda/\mu$, where $\lambda = \sum_{j=1}^{N} \lambda_j/N$ is the average update rate and $\mu = \sum_{j=1}^{N} \mu_j/N$ is the average access rate.

In the second-level adaptation, when an object is updated at the server, it is sent directly to the client if the object size is smaller than a threshold, called Push Threshold (T); otherwise, an invalidation message is sent to the client. The proposed adaptive scheme is shown in Fig. 1 and is explained as follows: In Step 1.1, when an access to one data object happens, the UAR information is updated so that the access to this object is taken into consideration. If it is a cache hit and the object is valid, the object is used; otherwise, Step 1.2 is executed to request the object from the server. In Step 1.3, the server sends back the object, in Step 1.4, the object is used, and based on the UAR value, U-threshold, and replacement policy, three issues are decided: 1) cache size K, 2) whether to cache this object or not, and 3) whether to evict another object from the cache or not. How to process 1, 2, and 3 will be discussed in later

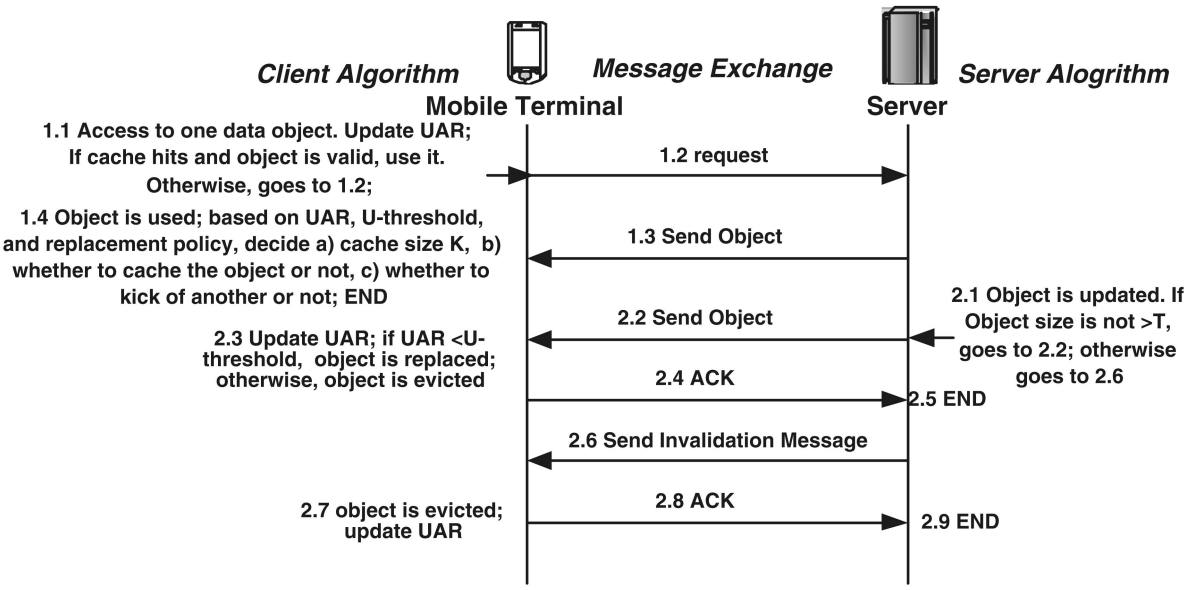

Fig. 1. Adaptive CB scheme.

sections. On the other hand, when an object is updated at the server (Step 2.1), if the object size is smaller than the Push threshold T, the updated object is sent back the client in Step 2.2; otherwise, an invalidation message is sent to the client in Step 2.6. In both Step 2.3 and Step 2.7, the UAR information is updated so that the update to this object is taken into consideration. In Step 2.3, if $UAR < U\text{-threshold}$, the object is replaced; otherwise, the object is evicted. In Step 2.7, the object is evicted.

If the sizes of all objects are larger than T, the second level adaptation scheme in Fig. 1 is equivalent to the Invalidation scheme. On the other hand, if the sizes of all objects are not larger than T, the second level adaptation scheme in Fig. 1 is equivalent to the Push scheme. The proposed scheme is an adaptive scheme between the Invalidation scheme and the Push scheme from this aspect. Our goal is to find an optimal threshold T to minimize the total traffic involved between the server and the client per data object access. It is clear that T should be not smaller than the size of the invalidation message.

In the mean time, another goal is to minimize the total traffic involved between the server and the client per data object access with an optimal cache size, based on the UAR. If the UAR is very large, we need to use the Fetch scheme.

Our major goal is to minimize the total traffic involved between the server and the client per data object access with an optimal cache size K and an optimal T value simultaneously, based on the value U in later sections.

3 Analytic Models

Lin et al. [5] propose analytical models for the Invalidation scheme and Poll-Each-Read scheme. Based on [5], we obtain an analytic model for the proposed adaptive CB cache mechanism under the Least Recently Used (LRU) replacement policy in Section 3.1. The proposed data access schemes are different from those in [5] so that analytical models are different. Then, we formulate a cost function as the total traffic involved between a server and a client in a mobile device per data object access in Section 3.2. We simplify the cost function under the assumption that object updates follow Poisson distribution in Section 3.3. This assumption will be loosened in the simulation Section 5.6, in which we adopt a cutoff Zipf-like update distribution [17], [18]. In Section 3.4, we study two special cases of the call scheme, i.e., the Push scheme and the Invalidation scheme. Finally, we study the cost when cache is disabled, i.e., the Fetch scheme, and give an approach to derive the $U\text{-threshold}$.

3.1 An Analytic Model

We have following assumptions:

- Sizes of mobile objects follow a density distribution $g(x)(\mathrm{x} > 0)$.
- A client can hold K objects in its local cache regardless object sizes. This assumption is loosened in simulations in Section 5.7.
- Accesses to a data object $O_k(k = 1, \ldots, N)$ follow the Poisson distribution with rate μ_k, where $\mu_k = \mu(k = 1, \ldots, N)$.
- Interarrival times between updates of a data object $O_k(k = 1, \ldots, N)$ follow a general distribution with the density function $f_k(t)$ with the mean $1/\lambda_k$, and the Laplace transform $f_k^*(s) = \int_{t=0}^{\infty} f_k(t)e^{-st}dt$, where λ_k=λ and $f_k(\mathrm{t}) = f(\mathrm{t})$ for $(k = 1, \ldots, N)$.

A data object O_i in the cache of the client is moved to the top of the cache when O_i is accessed. Consider Fig. 2, where the current access to O_i occurs at time t_4, the previous access to O_i occurs at time t_0, the previous update of O_i occurs at time t_1, and the next update of O_i occurs at time t_7. Consider another object O_j in Fig. 2, where the previous access to O_j occurs at time t_3, the next access to O_j occurs at time t_5, the previous update of O_j occurs at time t_2, and the next update of O_j occurs at time t_6. We have $\min\{t_0, t_1, t_2, t_3\} < t_4 < \min\{t_5, t_6, t_7\}$. Therefore, τ_0 and τ_3 have exponential distributions with mean $1/\mu$, and τ_1 and τ_2 have same general distributions with mean $1/\lambda$.

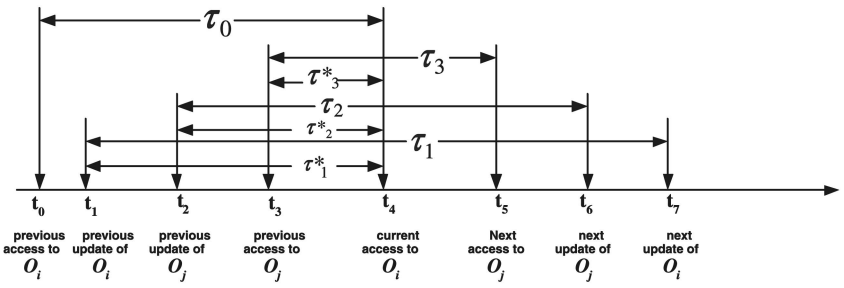

Fig. 2 Access and update diagram.

Since data access to O_i at t_4 is a random observer to τ_3 (property of the Poisson process), τ_3^* has the same distribution as τ_3 based on the excess life theorem [14] and the memoryless property of the exponential distribution. Furthermore, since data access to O_i at t_4 is a random observer to τ_1 and τ_2 (property of the Poisson process), based on the excess life theorem [14], τ_1^* and τ_2^* have density functions $r_{\tau_1^*}(t)$ and $r_{\tau_2^*}(t)$, respectively, where $r_{\tau_1^*}(t) = r_{\tau_2^*}(t) = \lambda \int_{x=t}^{\infty} f(x)dx$ and their Laplace transforms $r^*_{\tau_1^*}(s) = r^*_{\tau_2^*}(s) = \frac{\lambda}{s}[1 - f^*(s)]$; τ_1^* and τ_2^* have distribution functions $R_{\tau_1^*}(t)$ and $R_{\tau_2^*}(t)$, respectively, where $R_{\tau_1^*}(t) = R_{\tau_2^*}(t) = R(t)$.

After the access at t_0, O_i is stored in the cache with Rank 1 due to LRU. The cache ranking of O_i is not affected by an access to O_j during τ_0 if O_j is invalidated after the access; otherwise, the cache ranking of O_i is affected by an access to O_j. O_j is invalidated after an access to O_j if O_j's object size (L) is larger than the Push Threshold (denoted as T) and $t_3 < t_2$ (the same as $\tau_3^* > \tau_2^*$). In other words, the cache ranking of O_i is affected by an access to O_j during τ_0 if one of the following conditions happens: 1) $t_3 > t_2$ (the same as $\tau_3^* < \tau_2^*$) or 2) if O_j's object size (L) is not larger than T and $t_3 < t_2$ (the same as $\tau_3^* > \tau_2^*$). Let $\beta(t)$ be the probability that the cache ranking of O_i is affected by an access to O_j under the condition $\tau_0 = t$. Based on the derivation in Appendix A, we have

$$\beta(t) = 1 - e^{-\mu t} - \Pr(L > T) \int_{\tau_3^*=0}^{t} \mu e^{-\mu\tau_3^*} R(\tau_3^*) d\tau_3^*. \quad (1)$$

Let $P_{Adaptive\_CB}$ be the probability that an effective cache hit occurs at time t_4 when the adaptive CB scheme is adopted, respectively. The cached object O_i is not invalidated during period τ_0 if one of the following two conditions happens: 1) $\tau_1^* > \tau_0$ or 2) $\tau_1^* < \tau_0$ and O_i's object size (L) is not larger than T. Based on the derivation in Appendix A, we have

$$P_{Adaptive\_CB} = \sum_{k=0}^{K-1} \binom{N-1}{k} \int_{\tau_0=0}^{\infty} [\beta(\tau_0)]^k [1 - \beta(\tau_0)]^{N-k-1} \mu e^{-\mu\tau_0} [1 - \Pr(L > T) R(\tau_0)] d\tau_0. \quad (2)$$

3.2 Cost Function

In Fig. 1, the sizes of the request message in Step 1.2, the acknowledgement messages in Step 2.4 and 2.8, and the invalidation message in Step 2.6 are denoted as L_{Req}, L_{ACK}, and L_{Inv}, respectively. Their values are fixed values. An object size L is a variable value. In Step 1.3, there is no restriction on the object size L. In Step 2.2, the object size (L) is not larger than the push threshold, i.e., $L \leq T$. Let $M(L)$ be a cost when the message size is L. Let $E(U)$ denote the average number of updates of an object per data access to this object. Let $E(L)$ denote the mean value of object sizes. Let $E(L_{low})$ denote the mean value of object sizes that are smaller than T. Let $E(U)$ denote the average number of updates of an object per data access to this object. We formulate the cost function $C_{Adpative-CB}$ per access event as follows:

$$\begin{aligned} C_{Adpative-CB} &= (1 - P_{Adaptive\_CB})\big[M(L_{\text{Req}}) + M(E(L))\big] \\ &+ E(U)\big\{ \Pr(L > T)[M(L_{Inv}) + M(L_{ACK})] \\ &+ \Pr(L \leq T)[M(E(L_{low})) + M(L_{ACK})]\big\}. \end{aligned} \quad (3)$$

3.3 Poisson Update Distribution

In this section, we further assume that updates of a data object $O_k (k = 1, \ldots, N)$ follow a Poisson distribution with rate $\lambda_k = \lambda (k = 1, \ldots, N)$. Note that this assumption is loosened in the simulation Section 5.6, in which we adopt a cutoff Zipf-like update distribution [17], [18]. Define *update-to-access ratio* (UAR) as $U = \lambda/\mu$. Similar to [15], [16], we have

$$E(U) = U. \quad (4)$$

Then, τ_1^* and τ_2^* become exponential distributions, and $R(x) = [1 - e^{-\lambda x}]$. Equations (1) and (2) become (5) and (6), respectively:

$$\begin{aligned} &\beta(t) \\ &= 1 - e^{-\mu t} - \Pr(L > T) \int_{\tau_3^*=0}^{t} \mu e^{-\mu\tau_3^*} R(\tau_3^*) d\tau_3^* \\ &= 1 - e^{-\mu t} - \Pr(L > T) \int_{\tau_3^*=0}^{t} \mu e^{-\mu\tau_3^*} [1 - e^{-\lambda\tau_3^*}] d\tau_3^* \\ &= 1 - e^{-\mu t} - \Pr(L > T) \left[\left(1 - e^{-\mu t}\right) - \frac{\mu}{\mu + \lambda} \left(1 - e^{-(\mu+\lambda)t}\right) \right] \\ &= \Pr(L \leq T)\left(1 - e^{-\mu t}\right) + \Pr(L > T) \frac{1}{1 + U} \left(1 - e^{-(\mu+\lambda)t}\right), \end{aligned} \quad (5)$$

$$P_{Adaptive\_CB} = A(T) - \Pr(L > T) B(T), \quad (6)$$

where

$$A(T) \equiv \sum_{k=0}^{K-1} \binom{N-1}{k} \int_{\tau_0=0}^{\infty} [\beta(\tau_0)]^k [1 - \beta(\tau_0)]^{N-k-1} \mu e^{-\mu\tau_0} d\tau_0, \quad (7)$$

$$B(T) \equiv \sum_{k=0}^{K-1} \binom{N-1}{k} \int_{\tau_0=0}^{\infty} [\beta(\tau_0)]^k [1-\beta(\tau_0)]^{N-k-1} \mu e^{-\mu\tau_0} \left(1-e^{-\lambda\tau_0}\right) d\tau_0. \tag{8}$$

Define $M(x) = x$, i.e., the cost of a message is the size of the message. Equation (3) becomes (9):

$$C_{Adpative-CB} = [1 - A(T) + \Pr(L > T)B(T)]\left(L_{\text{Req}} + E(L)\right) + U[L_{Inv}\Pr(L > T) + E(L_{low})\Pr(L \le T) + L_{ACK}]. \tag{9}$$

Our goal is to minimize $C_{Adpative-CB}$ with both cache size K the Push threshold T.

3.4 Extreme Cases: Push and Invalidation

In this section, we consider two extreme/special cases. Note that if $L > T$ holds for all the time, the adaptive CB scheme becomes the Invalidation scheme; if $L \le T$ holds for all the time, the adaptive CB scheme becomes the Push scheme; and the adaptive CB scheme is an adaptive scheme based on the object size between the Invalidation scheme and the Push scheme. Let $\beta_{Inv}(t)$ and $\beta_{Push}(t)$ denote $\beta(t)$ values for the Invalidation scheme and the Push scheme, respectively. Let P_{Inv} and P_{Push} denote effective hit ratio values for the Invalidation scheme and the Push scheme, respectively. Let C_{Inv} and C_{Push} denote costs values per data object access for the Invalidation scheme and the Push scheme, respectively. We have

$$\beta_{Push}(t) = \left(1 - e^{-\mu t}\right). \tag{10}$$

Based on the derivation in Appendix B, we have

$$P_{Push} = \frac{K}{N}. \tag{11}$$

We have

$$C_{Push} = \left(1 - \frac{K}{N}\right)\left[L_{\text{Req}} + E(L)\right] + U[E(L_{low}) + L_{ACK}], \tag{12}$$

$$\beta_{Inv}(t) = \frac{\mu}{\mu+\lambda}\left(1 - e^{-(\mu+\lambda)t}\right). \tag{13}$$

Based on the derivation in Appendix B, we have

$$P_{Inv} = \frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N}, \tag{14}$$

$$C_{Inv} = \left(1 - \frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N}\right)\left[L_{\text{Req}} + E(L)\right] + U(L_{Inv} + L_{ACK}). \tag{15}$$

3.5 Fetch and U-Threshold

For the Fetch scheme, the cost function is very simple and is given as follows:

$$C_{Fetch} = L_{\text{Req}} + E(L). \tag{16}$$

The U-threshold is defined as a U value when the Fetch scheme is better than the adaptive CB scheme in (9). Therefore, when U is larger than U-threshold, the Fetch scheme is adopted, i.e., the cache is disabled or the cache size is zero. To obtain U-threshold, we can numerically solve the equation: $C_{Fetch} = C_{Adpative-CB}$.

4 Optimality Analysis

In this section, we conduct optimality analysis for the proposed scheme. In Section 4.1, we study optimality of the Push threshold. In Section 4.2, we study optimality of cache size as functions of $U = \lambda/\mu$. Finally, in Section 4.3, we discuss how to optimize both the cache size and the Push threshold simultaneously.

4.1 Optimal Push Threshold

We study a constant distribution, a uniform distribution, and an exponential distribution for data object sizes one by one. Note that other distributions of data object sizes can be pursued similarly.

4.1.1 Data Object: Same Size

In this section, we assume that the sizes of data objects are all the same. Under such an assumption, we will try to find the optimal T value to minimize $C_{Adpative-CB}$. The density function has the following form:

$$g(x) = \begin{cases} 1, x = L \\ 0, \text{otherwise.} \end{cases} \tag{17}$$

Since both L and T are fixed values, we have either $\Pr(L < T) = 0$ or $\Pr(L < T) = 1$ all the time. If $L > T$, this is the Invalidation scheme; otherwise, if $L \le T$, this is the Push scheme. To choose the optimal T, we can let $C_{Push} = C_{Inv}$ using (12) and (15). We have $E(L) = E(L_{low}) = T$ and

$$T_{Optimal} = \left[UL_{Inv} + \frac{K}{N}L_{\text{Req}} - L_{\text{Req}}\frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N}\right] \Bigg/ \left[U - \frac{K}{N} + \frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N}\right]. \tag{18}$$

Assuming that $L_{Inv} = L_{\text{Req}}$, we have

$$T_{Optimal} = L_{Inv}\left[U - \frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N} + \frac{K}{N}\right] \Bigg/ \left[U + \frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N} - \frac{K}{N}\right]. \tag{19}$$

We have two intuitions as follows: First, as stated before, when an object size is not larger than the invalidation message size, then the object should be sent instead of the invalidation message. In other words, we should have $T_{Optimal} \geq L_{Inv}$. Second, when the update rate is extremely high and the access rate is small, a data object in cache is more likely to be obsolete very often. Under such a condition, the replaced objects have less usage, and the scheme merges to the Invalidation scheme. Therefore, we should have $\lim_{U\to\infty} T_{Optimal} = L_{Inv}$. We have the following lemma, and the proof is shown in Appendix C:

Lemma 1. *If we assume that $L_{Inv} = L_{\mathrm{Req}}$ holds and data objects are all the same size, we have $T_{Optimal} \geq L_{Inv}$ and $\lim_{U\to\infty} T_{Optimal} = L_{Inv}$.*

However, in our proposed optimal callback with two-level adaptation scheme, when U is large enough, the Fetch scheme should be used. In the above lemma, we did not consider the cache size and, therefore, the Fetch scheme is not considered.

Note that T is the threshold which is independent of the data object size, and therefore, T is fixed when data objects have the same size.

4.1.2 Data Object Size: Uniform Distribution

Assume that data object sizes follow a uniform distribution as follows:

$$g(x) = \begin{cases} \frac{1}{2a}, x \in [E(L)-a, E(L)+a] \\ 0, \text{otherwise.} \end{cases} \tag{20}$$

Since $T < E(L) - a$ is meaningless and $T > E(L) + a$ is equivalent to $T = E(L) + a$, we only consider $T \in [E(L) - a, E(L) + a]$. Then, we have

$$\Pr(L \leq T) = \frac{T - E(L) + a}{2a}, \tag{21}$$

$$\Pr(L > T) = \frac{E(L) + a - T}{2a}, \tag{22}$$

$$E(L_{low}) = \frac{T + E(L) - a}{2}. \tag{23}$$

Plugging (21)-(23) into (5)-(9), we have

$$\begin{aligned} &C_{Adpative-CB} \\ &= \left\{1 - A(T) + B(T)\frac{E(L)+a-T}{2a}\right\}\left(L_{\mathrm{Req}} + E(L)\right) + U \\ &\left[L_{Inv}\frac{E(L)+a-T}{2a} + \frac{T+E(L)-a}{2}\frac{T-E(L)+a}{2a} + L_{ACK}\right]. \end{aligned} \tag{24}$$

We obtain the first derivative of (24) as follows:

$$\begin{aligned} &\frac{\partial C_{Adpative-CB}}{\partial T} \\ &= \left[-\frac{\partial A(T)}{\partial T} - \frac{B(T)}{2a} + \frac{E(L)+a-T}{2a}\frac{\partial B(T)}{\partial T}\right]\left(L_{\mathrm{Req}} + E(L)\right) \\ &+ U\left[-L_{Inv}\frac{1}{2a} + \frac{1}{2}\frac{T-E(L)+a}{2a} + \frac{T+E(L)-a}{2}\frac{1}{2a}\right]. \end{aligned} \tag{25}$$

If we let $\frac{\partial C_{Adpative-CB}}{\partial T} = 0$, we can obtain the optimal threshold $T_{Optimal}$ explicitly.

4.1.3 Data Object Size: Exponential Distribution

Assume that data object size follows an exponential distribution as follows:

$$g(x) = \begin{cases} \gamma e^{-\gamma x}, x > 0 \\ 0, \text{otherwise.} \end{cases} \tag{26}$$

Since considering $T < 0$ is meaningless, we only consider $T \geq 0$. Then, we have

$$\Pr(L \leq T) = \int_{x=0}^{T} \gamma e^{-\gamma x} dx = 1 - e^{-\gamma T}, \tag{27}$$

$$\Pr(L > T) = \int_{x=T}^{\infty} \gamma e^{-\gamma x} dx = e^{-\gamma T}, \tag{28}$$

$$E(L) = \frac{1}{\gamma}, \tag{29}$$

$$E(L_{low}) = \int_{x=0}^{T} \gamma x e^{-\gamma x} dx = \frac{1}{\gamma} - Te^{-\gamma T} - \frac{1}{\gamma}e^{-\gamma T}. \tag{30}$$

Plugging (27)-(30) into (5)-(9), we have

$$\begin{aligned} C_{Adpative-CB} &= \left(1 - A(T) + e^{-\gamma T}B(T)\right)\left(L_{\mathrm{Req}} + \frac{1}{\gamma}\right) \\ &+ U\left[L_{Inv}e^{-\gamma T} + \left(\frac{1}{\gamma} - Te^{-\gamma T} - \frac{1}{\gamma}e^{-\gamma T}\right)\left(1 - e^{-\gamma T}\right) + L_{ACK}\right]. \end{aligned} \tag{31}$$

We obtain the first derivative of (31) as follows, while the derivation is shown in Appendix D:

$$\begin{aligned} &\frac{\partial C_{Adpative-CB}}{\partial T} = \\ &\left[-\frac{\partial A(T)}{\partial T} - \gamma e^{-\gamma T}B(T) + e^{-\gamma T}\frac{\partial B(T)}{\partial T}\right]\left(L_{\mathrm{Req}} + \frac{1}{\gamma}\right) + U \\ &\left[-\gamma L_{Inv}e^{-\gamma T} + e^{-\gamma T} - Te^{-2\gamma T} - \frac{1}{\gamma}e^{-2\gamma T} + \gamma Te^{-\gamma T} - \gamma Te^{-2\gamma T}\right]. \end{aligned} \tag{32}$$

If we let $\frac{\partial C_{Adpative-CB}}{\partial T} = 0$, we can obtain the optimal threshold $T_{Optimal}$ numerically.

4.2 Optimal Cache Size

The cache size K has a finite range [0, M], where M is the maximum physical cache size in an MT. For a given Push threshold and a given UAR, we can obtain the optimal K value by comparing M+1 values of the cache size.

One extreme case is that when the UAR is very large, the Fetch scheme should be used since objects are always obsolete. Note that the Fetch scheme can be treated as a special case when the cache size is zero. Another extreme case is that when the UAR is zero, the cache size should be M. In other words, the optimal K value ($K_{Optimal}(T, U)$) is a function ($F(T, U)$) of the Push threshold and UAR as follows:

$$K_{Optimal}(T,U) = F(T,U) = \begin{cases} 0, & U \geq U_{Threshold} \\ M, & U = 0 \\ F(T,U), & otherwise. \end{cases} \tag{33}$$

The function F can be achieved by comparing the finite number of cases.

4.3 Optimizing Both the Cache Size and the Push Threshold Simultaneously

The major goal of this paper is to show that we can optimize T and K at the same time. We adopt the following steps:

Step 1: For each fixed K value within [0, M], obtain the optimal cost $C_{Optimal}(K)$ with the optimal Push threshold $T_{Optimal}(K)$.

Step 2: Compare the above M+1 values to find the minimum cost $C_{Optimal}$ and corresponding $T_{Optimal}$ and $K_{Optimal}$. We have

$$C_{Optimal} = \min_{0 \le K \le M} \{C_{Optimal}(K)\}, \tag{34}$$

$$T_{Optimal} = T_{Optimal}(k_1), \quad \text{where } C_{Optimal}(k_1) = \min_{0 \le K \le M} \{C_{Optimal}(K)\}, \tag{35}$$

$$K_{Optimal} = k_1, \text{where } C_{Optimal}(k_1) = \min_{0 \le K \le M} \{C_{Optimal}(K)\}. \tag{36}$$

4.4 Comments on U-Threshold

In Section 3.5, we explain how to obtain U-threshold. An alternative way of obtaining it is described as follows: U-threshold is a threshold that, when the UAR value is larger than the threshold, the cache is disabled. In another words, the cache size is 0 when the UAR is larger than U-threshold. According to (19), (25), and (32), T-thresholds can be estimated with a given K. By using the estimated T-thresholds, we can compute the cost at any K and UAR. We compare the cost for all the cases when $K = 0, 1, \text{ to } M$ by given a UAR. Therefore, we know the K value for a given UAR. The least UAR that makes K equal to 0 will be U-threshold.

5 Performance Evaluation

In this section, we conduct performance evaluation for all the schemes. In Section 5.1, we conduct simulations to verify analytical results. We study the optimal Push Threshold in Section 5.2. We study optimal cost when the cache size and Push threshold are optimized simultaneously in Section 5.3. In Section 5.4, we study the U-threshold. In Section 5.5, we study the impact of the size of N. In Section 5.6, we will loosen the assumption that the object updates have the same ratio. Furthermore, we will loosen the assumption that the update process follows Poisson distribution, and instead, we adopt a cutoff Zipf-like update distribution [17], [18], which is believed to be a realistic distribution. Finally, in Section 5.7, we further loosen the assumption that a client can hold K objects in its local cache regardless object sizes in our simulations. Note that in Sections 5.1, 5.5, 5.6, and 5.7, results are simulation results, and in Sections 5.1, 5.2, 5.3, and 5.4, results are numerical results of analytical models.

Define *update-to-access ratio* (UAR) as $U = \lambda/\mu$, where λ is the update rate and μ is the access rate. Let M denote the maximum physical size of an MT, where $0 \le K \le M \le N$,

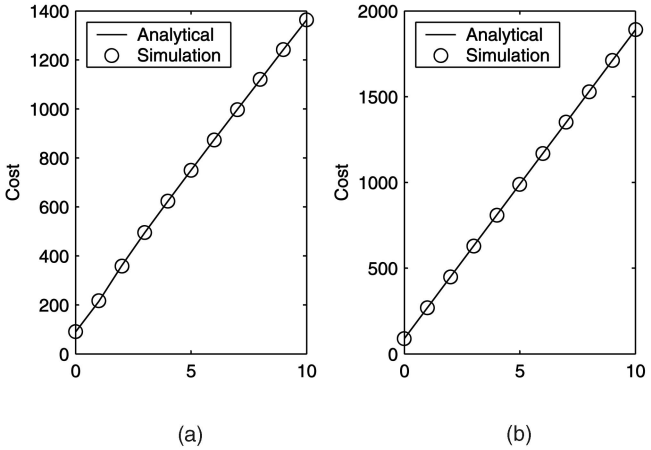

Fig. 3. Simulation results versus analytical results. (a) Cost versus U (Invalidation). (b) Cost versus U (Push).

and K is the cache size and N is the total number of objects at the server. L denotes the object size, and $E(L)$ denotes the mean of L. T denotes the Push threshold and U-threshold denotes the UAR threshold. We have the following parameters unless otherwise stated: $N = 100$, $M = 70$, $K = 70$, $\mu = 1$, and $L_{Inv} = L_{\text{Req}} = L_{ACK} = 30$.

5.1 Simulation Validations

Fig. 3 shows the costs of simulations and analytic results where $L_{Inv} = L_{\text{Req}} = L_{ACK} = E(L)/2 = 60$ and $K = 50$. The simulation adopts discrete event simulation using C++. Fig. 3 indicates that analytical results match simulation results exactly for both the Push scheme and the Invalidation scheme.

5.2 Optimal Push Threshold

We study a constant distribution, a uniform distribution, and an exponential distribution for object sizes in this section.

Figs. 4a, 4b, and 4c show the costs under a constant distribution, a uniform distribution, and an exponential distribution, respectively, where $E(L) = 30$. Fig. 4a shows the cost over the object size, and Figs. 4b and 4c show the cost over the Push threshold. Figs. 4a, 4b, and 4c all indicate that when U is larger, the cost is higher, since there are more updates. Fig. 4a shows that when the object size is small, the Push scheme is better, and when the object size is large, the Invalidation scheme is better. Figs. 4b and 4c show that the cost decreases as the threshold T increases when T is small, and increases as T increases when T is large. Therefore, an optimal T value exists for both Figs. 4b and 4c. Figs. 4b and 4c also show that the shapes of curves are similar under uniform and exponential distributions of object sizes.

In Fig. 4a, a larger U value makes the cost caused by data object updates weigh more than the cost caused by data object accesses, and therefore, causes either the Push scheme or the Invalidation scheme to perform better than another based on the object size. With some simple calculations, $(C_{Push} - C_{Inv})/C_{Inv}$ could be as high as ($\pm$) 30~40 percent. Therefore, a better design is necessary.

Fig. 5 shows $T_{Optimal}$ values over U under a constant distribution, a uniform distribution, and an exponential

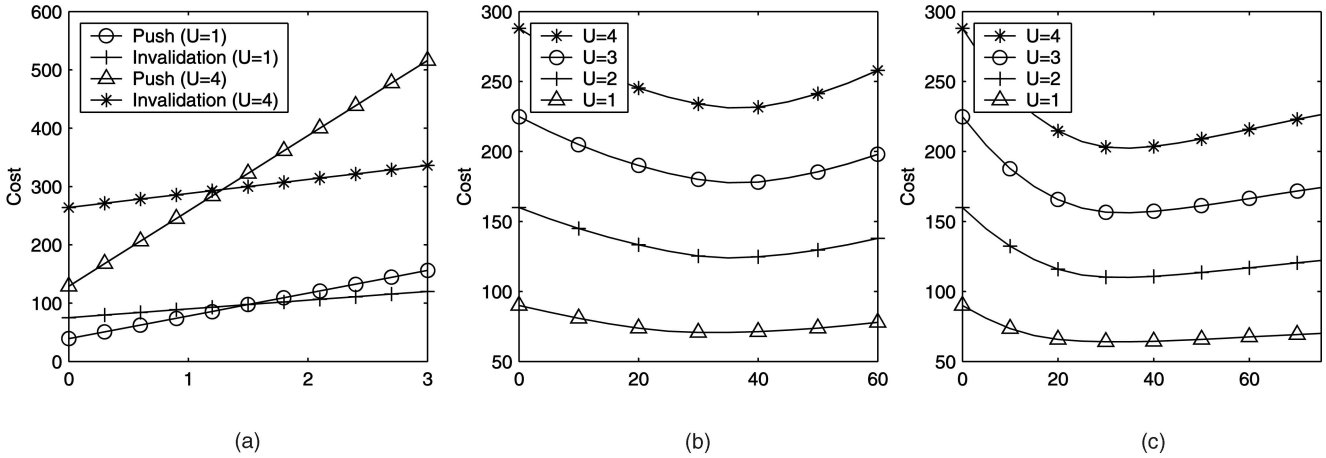

Fig. 4. Cost. (a) Cost versus $\mathrm{L}/\mathrm{L}_{\mathrm{inv}}$ (Constant). (b) Cost versus T (Uniform). (c) Cost versus T (Exponential).

distribution, respectively, where $T_{Optimal}$ is the optimal Push threshold and $E(L) = 30$. Fig. 5a shows that when U is not very large (<1.5), $T_{Optimal}$ increases as U increases, and when U is large, $T_{Optimal}$ decreases slowly as U increases since a data object in cache is more likely to be obsolete very often and the replaced objects have less usage. For the uniform distribution and the exponential distribution, the curves (Figs. 5b and 5c) are similar to Fig. 5a, except that $T_{Optimal}$ changes more slowly with the U value.

Fig. 6 shows costs over the object size under a constant distribution, a uniform distribution, and an exponential distribution, respectively, where $U = 1$. For all distributions, the cost increases as the object size increases. Fig. 6a compares the costs of the Push scheme, the Invalidation scheme, and the proposed optimal adaptive CB scheme;

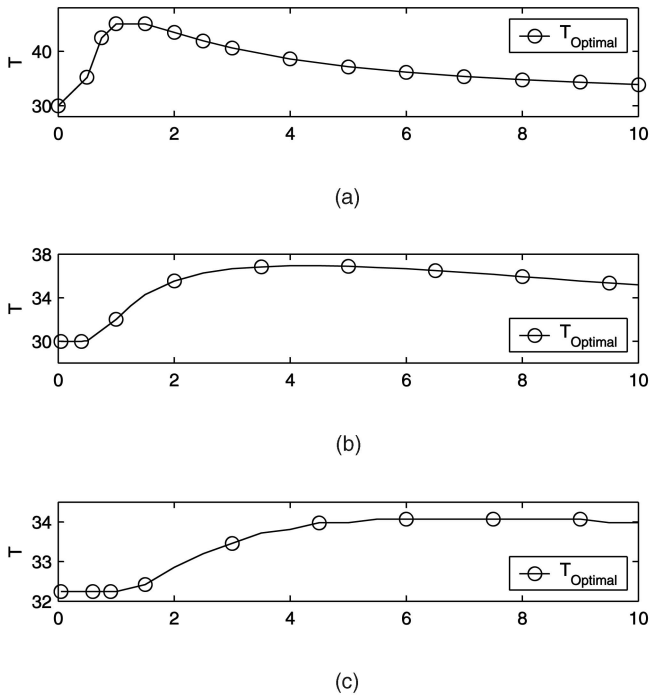

Fig. 5. $T_{Optimal}$. (a) T versus U (Constant). (b) T versus U (Uniform). (c) T versus U (Exponential).

when the object size is small, the Push scheme is better than the Invalidation scheme, and when object size is large, the Invalidation scheme is better than the Push scheme. The optimal adaptive scheme is the best scheme. Figs. 6b and 6c compare costs with different Push thresholds and the optimal Push threshold, and indicates that the optimal threshold achieves the minimum cost.

5.3 Optimize Both Cache Size and Push Threshold Simultaneously

In this section, we study optimizing both the cache size and Push threshold simultaneously.

Fig. 7 shows costs over U when the object size follows a constant distribution, a uniform distribution, and an exponential distribution, respectively, where $E(L) = 10L_{Inv}$ and $T = 5L_{Inv}$. Under the parameters chosen, the cache size $K = M$ is almost the best, except that when U is large, the no-cache scheme, the Fetch scheme, has a lower cost. As shown in the figure, we observe that when U is large, the cache should be disabled.

Fig. 8 compares the optimal cost with costs of the Push scheme and the Invalidation scheme when the object size follows a constant distribution, a uniform distribution, and an exponential distribution, where $E(L) = 10L_{Inv}$. The optimal cost is obtained when both the cache size and the Push threshold are optimized simultaneously. Fig. 8 indicates that the proposed scheme is the best among all the schemes in term of cost.

Fig. 9 shows the optimal cache size (Optimal K) (the Push threshold is also optimal) over U when the object size follows a constant distribution, a uniform distribution, and an exponential distribution. As illustrated from the figure, when $E(L) = L_{Inv}$ and $E(L) = 10L_{Inv}$, the optimal K changes from the maximum cache size (M = 70) directly into zero, i.e., the Fetch scheme. When $E(L) = 100L_{Inv}$, we observe that the optimal cache size changes slowly as U increases, and finally becomes zero when U is large. From this figure, we can conclude that the no-cache (Fetch) scheme is very important and should be used when U is large.

Fig. 10 shows the optimal T (the cache size is also optimal) over U when the object size follows a constant distribution, a uniform distribution, and an exponential distribution. As illustrated in the figures, the optimal T

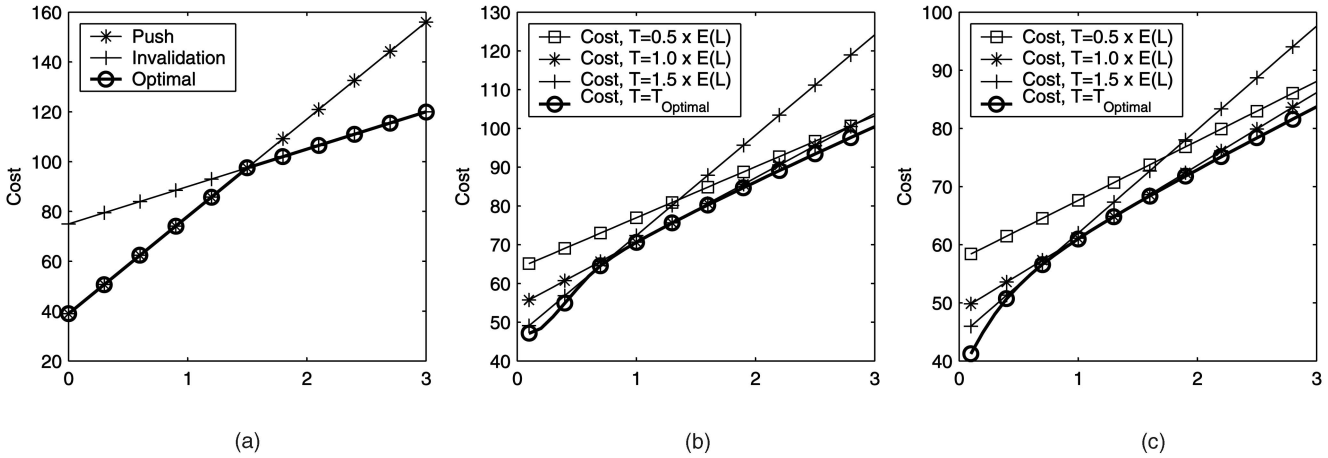

Fig. 6. Cost. (a) Cost versus $\mathrm{E(L)/L_{inv}}$ (Constant). (b) Cost versus $\mathrm{E(L)/L_{inv}}$ (Uniform). (c) Cost versus $\mathrm{E(L)/L_{inv}}$ (Exponential).

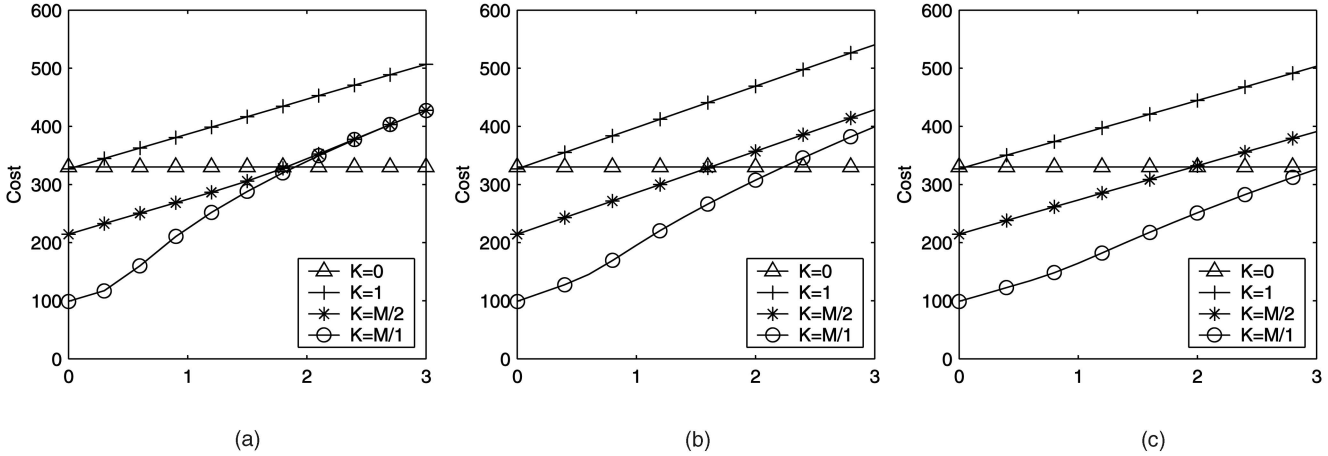

Fig. 7. Cost versus U with different K values. (a) Cost versus U (Constant). (b) Cost versus U (Uniform). (c) Cost versus U (Exponential).

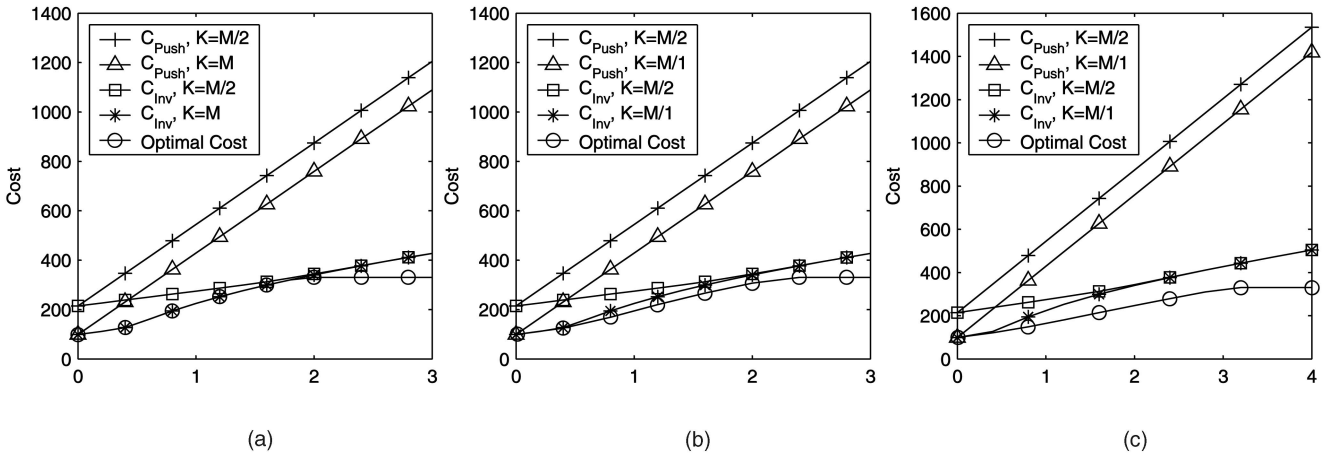

Fig. 8. Optimal cost, costs of Push, and Invalidation versus U. (a) Cost versus U (Constant). (b) Cost versus U (Uniform). (c) Cost versus U (Exponential).

(with an optimal K) increases first and then decreases as U increases. The curve shapes in Fig. 10 are similar to those in Fig. 5, i.e., the case when the optimal K is not considered. From the figure, we observe that at some cases, there are no optimal T values, since at those situations, the cache is disabled and the Fetch scheme is used.

5.4 U-Threshold

Fig. 11 shows the U-threshold over E(L) when the object size follows a constant distribution, a uniform distribution, and an exponential distribution. As illustrated in the figure, U-threshold increases as $E(L)$ increases. U-threshold is similar with different distributions of object sizes.

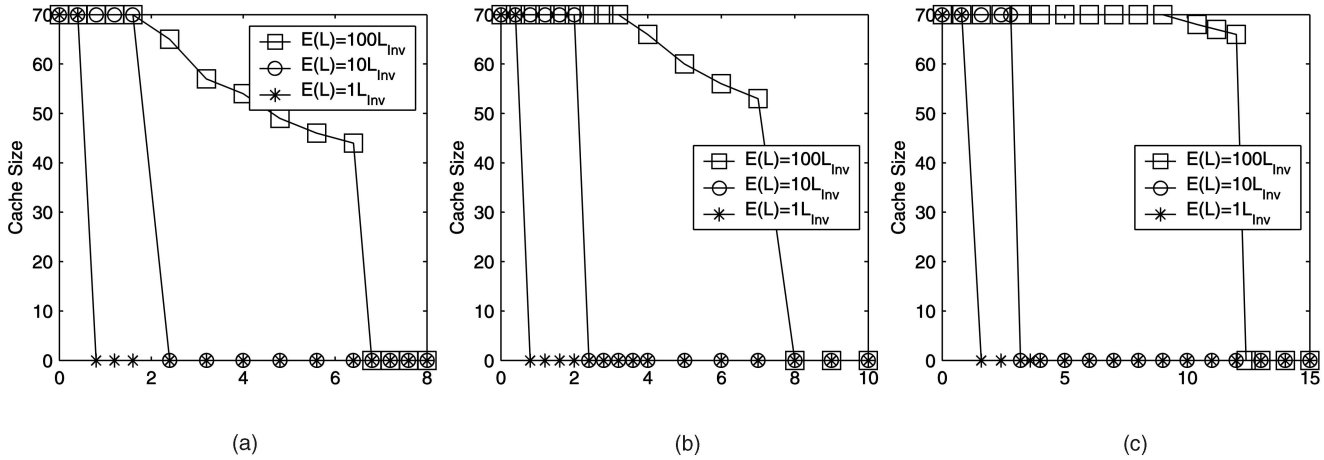

Fig. 9. Optimal K versus U (with Optimal T). (a) Cache size versus U (Constant). (b) Cache size versus U (Uniform). (c) Cache size versus U (Exponential).

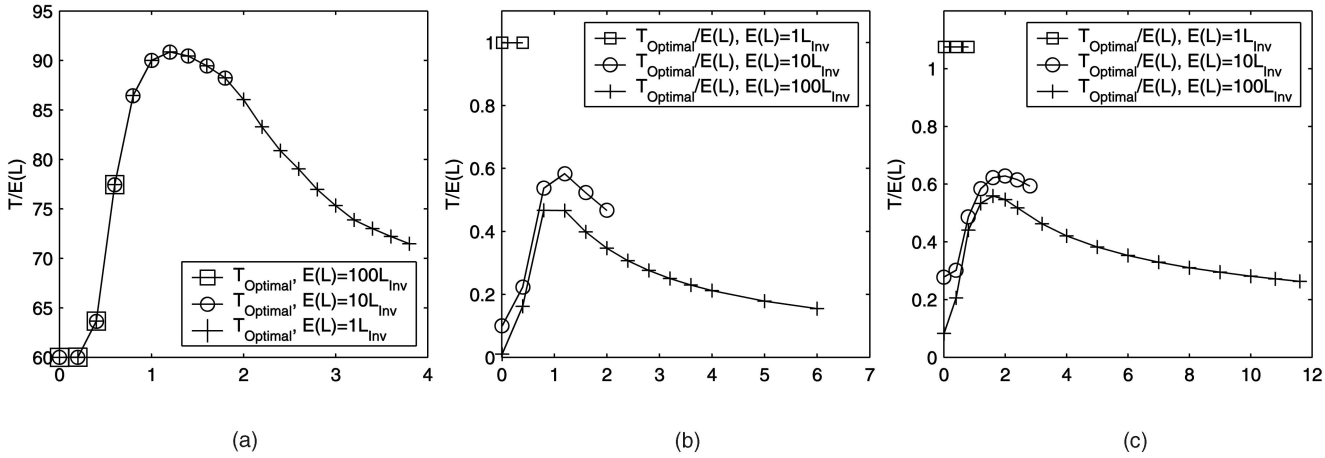

Fig. 10. Optimal T versus U (with Optimal K). (a) T/E(L) versus U (Constant). (b) T/E(L) versus U (Uniform). (c) T/E(L) versus U (Exponential).

5.5 Impact of the Size of N

In this section, we study the impact of the size of N. All the parameters are the same as before except that Fig. 12 has the following parameters: $E(L) = 120$ and $L_{req} = L_{inv} = L_{ack} = 60$, and Fig. 13 has the following parameters: $E(L) = 300$ and $L_{req} = L_{inv} = L_{ack} = 30$.

Figs. 12a, 12b, and 13 show the costs over the K/N ratio with the different UAR values of the Invalidation scheme, the Push, and the adaptive CB scheme, respectively. In the adaptive CB scheme, the T-threshold is used and the cache is not disabled. Furthermore, the uniform distribution and

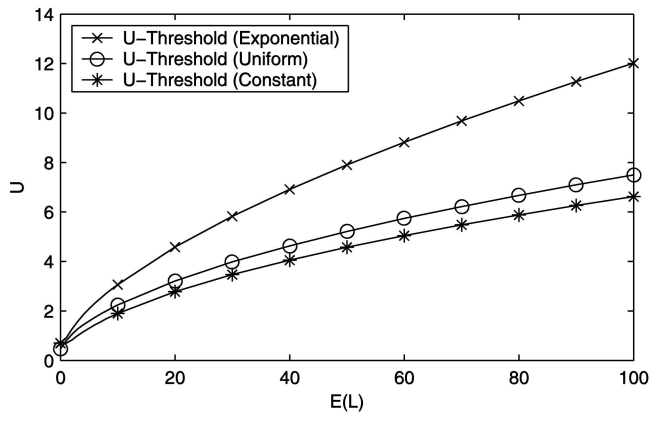

Fig. 11. U-Threshold.

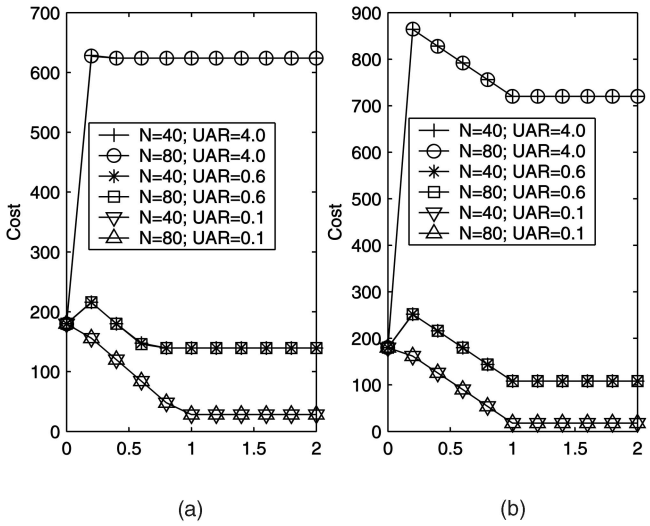

Fig. 12. Cost for invalidation and push. (a) Cost versus K/N (Invalidation). (b) Cost versus K/N (Push).

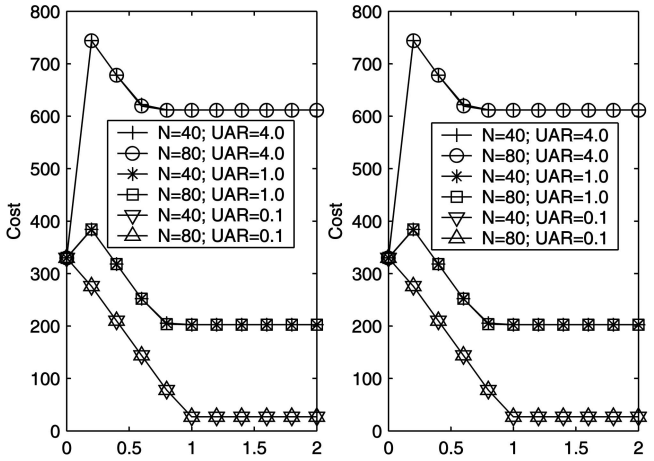

Fig. 13. Cost for the adaptive CB. (a) Cost versus K/N (Uniform). (b) Cost versus K/N (Exponential).

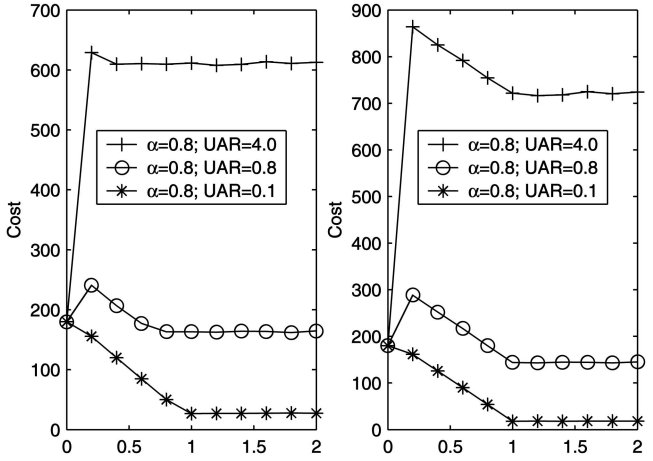

Fig. 14. Cost for Invalidation and Push with cutoff Zipf-like update distribution. (a) Cost versus K/N (Invalidation). (b) Cost versus K/N (Push).

exponential distribution of object sizes are shown in Figs. 13a and 13b, respectively. From Figs. 12 and 13, we observe that the costs are the same for different N values (either 40 or 80) as long as the K/N ratios are the same. It is clearly shown in Figs. 12 and 13 that when UAR is larger, the cost is larger. As illustrated in the figures, in general, when the cache size increases, the cost decreases; however, there are exceptions when UAR is large. Therefore, a larger cache size does not always mean a lower cost with a large UAR. When the cache size reaches a threshold, increasing the cache size makes no difference in terms of cost reduction.

5.6 A "Cutoff" Zipf-Like Update Distribution with Different Update Ratios

In this section, we loosen the assumption that the object updates have the same ratio, and we loosen the assumption that the update process follows Poisson distribution. Instead, we adopt a cutoff Zipf-like update distribution [5], [17], [18], which is believed to be a realistic distribution. The goal of this section is to see the performance of the the proposed scheme when these assumptions are loosened. When an update happens, the object O_i is chosen with a given probability p_k defined as

$$p_k = 1 \Big/ \left[k^{\alpha} \left(\sum_{j=1}^{N} \frac{1}{j^{\alpha}} \right) \right],$$

where α is called the Zipf ratio. In this paper, an update event is generated with mean $\lambda = \mu \cdot UAR$ and $\mu = 1$. In this section, we let $\alpha = 0.8$.

Next, we explain how to use the U-threshold and T-threshold in simulations when many assumptions are loosened. The previous analytic models compute the U-threshold and T-threshold with the assumption that UARs of all objects are the same. We calculate the U-threshold and T-threshold when given an object size and a UAR. Therefore, we obtain a table (omitted), in which we can look up the U-threshold and T-threshold given a frame size and a UAR value. Items in the table can be used as points (U-threshold, T-threshold) for the interpolation method to obtain other points.

When we lose the assumptions that the UARs of objects are the same and the object sizes are the same, there must exist U-threshold and T-threshold as well. However, we will show that using the U-threshold and T-threshold obtained from the previous analytic models, the algorithm performs quite well. In this case, for each object, we measure its object size and UAR. We look up the values of the U and T-thresholds from the table. If the object size and UAR not found from the table, we can use a linear interpolation to find an approximate one. As long as the table is dense enough, the two thresholds will be precise enough. If the measured UAR is larger than the U-threshold, the object will not be cached. When the UAR is smaller than the U-threshold, the corresponding T-threshold will be used to determine whether the Push scheme or the Invalidation scheme to be used.

Fig. 14 shows the cost for Invalidation and Push with cutoff Zipf-like update distribution to study the effects of the size of N. Fig. 14 has the following parameters: $E(L) = 120$ and $L_{req} = L_{inv} = L_{ack} = 60$. Comparing Fig. 14 with Fig. 12, we observe that with different update ratios and a cutoff Zipf-like update distribution, the effect is almost the same.

The cutoff Zipf-like update distribution only provides the update probabilities of different objects. However, when the sizes of objects are not the same, we need to consider how we define these probabilities to objects with different sizes, i.e., how to assign the update rates to objects with different sizes. For performance evaluation purpose, we define the following two cases. We define SOFU as a case that small objects are more frequently updated than large objects, and define LOFU as a case that large objects are more frequently updated than small objects. More specifically, we have the following steps.

- Calculate N Zipf-like probabilities $p_i, (i = 1, \ldots, N)$. Note that with a large rank, the probability is smaller.

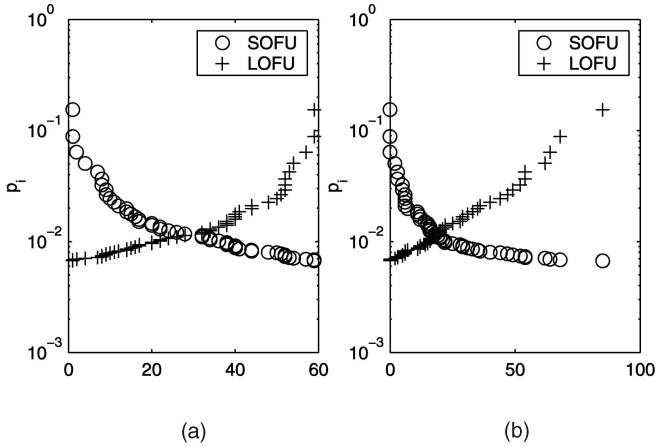

Fig. 15. Ranking objects in SOFU and LOFU. (a) p_i versus Size (Uniform). (b) p_i versus Size (Exponential).

- Generate N objects with the sizes following a uniform/exponential distribution.
- Match the probabilities with the objects according to their sizes. In other words, rank the objects so as to match the probabilities. When SOFU is used, rank the objects with the increasing order of the object sizes. When LOFU is used, rank the objects with the decreasing order of the object sizes.

Fig. 15 shows an example using SOFU and LOFU to match probabilities to object sizes. The object sizes are uniform distributed in $[0, 2 \times E(L)]$ or exponential distributed with the mean $E(L)$, where $N = 100$ and $E(L) = 300$.

Figs. 16, 17, and 18 have the following parameters: $N = 100$, $M = 70$, $L_{req} = L_{inv} = L_{ack} = 30$, and $\mu = 1.0$. Figs. 16, 17, and 18 correspond to Figs. 4, 7, and 8, respectively, but with different update ratios for different objects. Comparing these figures, we observe that Figs. 16, 17, and 18 exactly have the same effects as Figs. 4, 7, and 8, respectively. In other words, under a Zipf-like distribution with different update ratios, the conclusions obtained from the previous sections are still valid. Fig. 16 shows that there exists an optimal T threshold and SOFU has a smaller cost than LOFU. Fig. 17 shows that the U threshold exists when the second level adaptation. Fig. 18 shows optimization of both T and U at the same time.

5.7 Loosen the Assumption that the Client Can Hold K Objects Regardless of Object Sizes

In this section, we further loosen the assumption that a client can hold K objects in its local cache regardless object

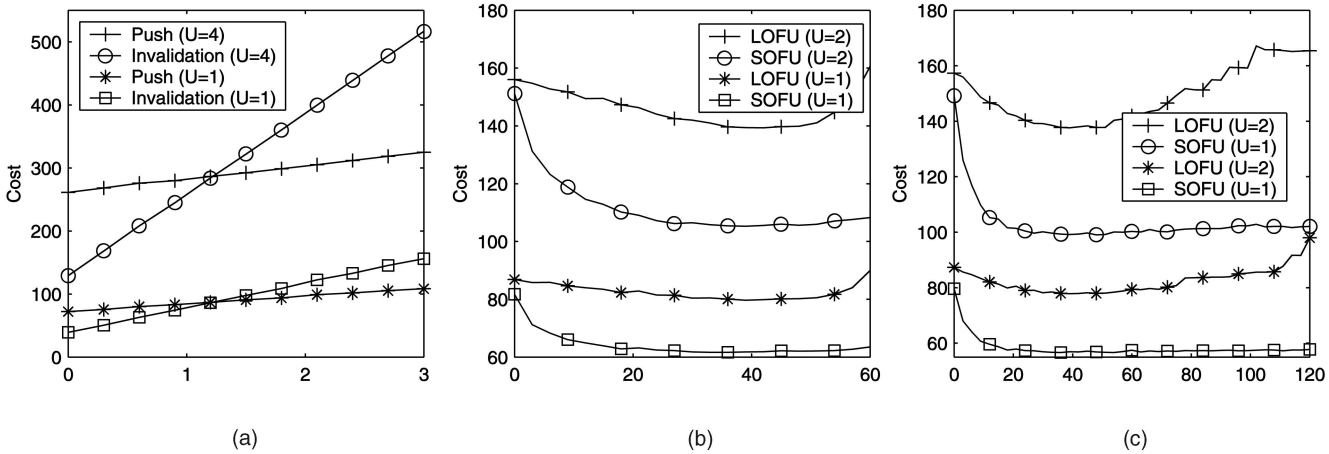

Fig. 16. Cost with different update ratios. (a) Cost versus L/L_{inv} (Constant). (b) Cost versus T (Uniform). (c) Cost versus T (Exponential).

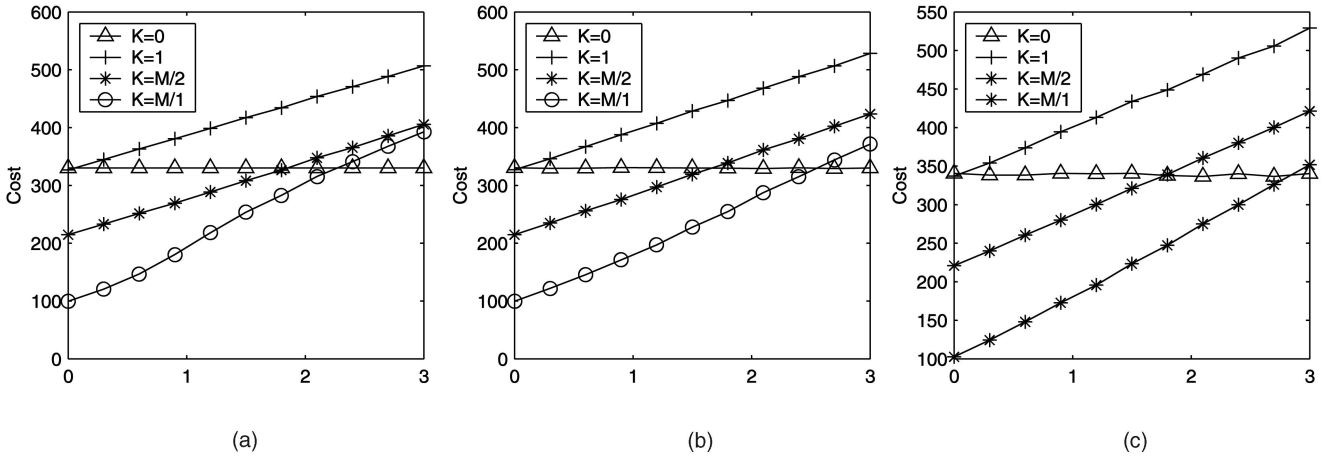

Fig. 17. Cost versus U with different K values and different update ratios. (a) Cost versus U (Constant). (b) Cost versus U (Uniform-SOFU). (c) Cost versus U (Exponential-SOFU).

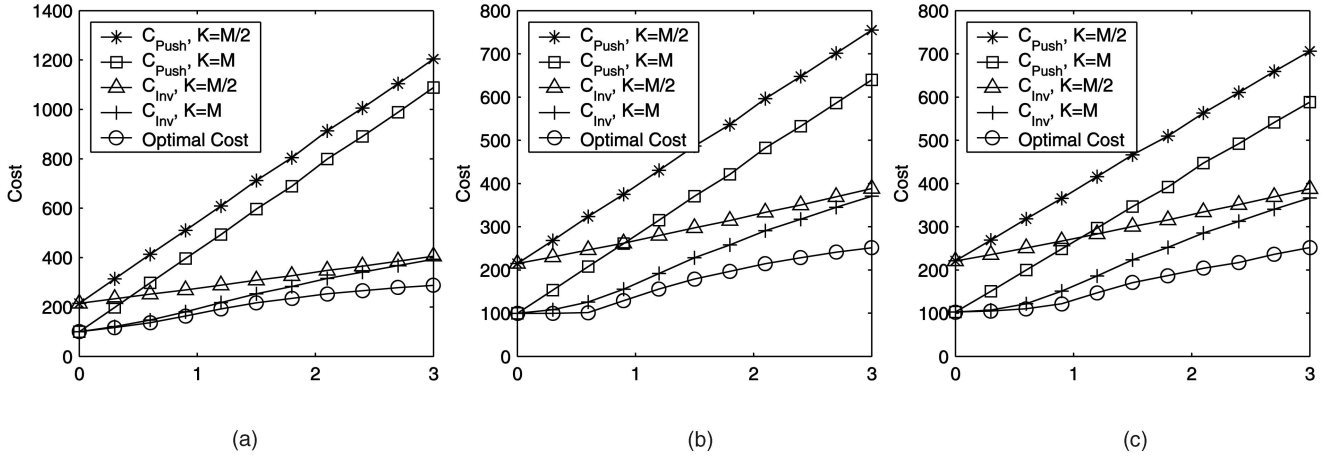

Fig. 18. Optimal cost, costs of Push and Invalidation versus U, with different update ratios. (a) Cost versus U (Constant). (b) Cost versus U (Uniform-SOFU). (c) Cost versus U (Exponential-SOFU).

sizes. The cache size is measured by bytes instead of K objects. The maximum cache size is taken at 0.7 * total length of all objects at the server. Since we use the LRU replacement policy, we evict one or more objects as long as there is enough free space in the cache to accommodate the object being accessed. Fig. 19 is to redraw Fig. 18b and Fig. 18c after loosening the assumption. Comparing Fig. 19 and Fig. 18, we observe that the similar results are obtained as previous figures do.

6 Conclusions

In this paper, we propose an *optimal callback with two-level adaptation* scheme for wireless data access. We analytically model a cost function as the total traffic involved between a server and an MT per data object access, and the optimal T value and the optimal cache size are obtained to minimize the cost function. An important threshold, called U-threshold, is defined as an *update-to-access-ratio* (UAR) value, beyond which the object should not be cached. We study a constant distribution, a uniform distribution, and an exponential distribution for object sizes, and we have results as follows:

- Analytical results match simulation results exactly.
- The cost is minimized with an optimal cache size and an optimal Push threshold.
- U-threshold increases as the UAR increases, and U-threshold is an important threshold and concept.
- As object size increases, costs of all schemes increase.
- The optimal T (with an optimal increase) increases first and then decreases as UAR increases
- For different distributions of object sizes, effects of the proposed algorithm are almost similar.
- The impact of the size of N has been studied. We observe that the costs are the same for different N values as long as K/N ratios are the same. In general, when the cache size increases, the cost decreases; however, there are exceptions when UAR is large. Therefore, a larger cache size does not always mean a lower cost with a large UAR. When the cache size reaches a threshold, increasing cache size makes no difference in terms of cost reduction.
- In our simulations, we further loosened the assumption that the update distribution is Poisson and considered a cutoff Zipf-like update distribution with different update ratios. The conclusion is that similar results are still valid.
- At last, in our simulations, we further loosen the assumption that a client can hold K objects in its local cache regardless object sizes, and we observe that the similar results are obtained.

In summary, the proposal for an adaptive access mechanism demonstrates that it has better performance than several traditional approaches and achieves the optimal solutions.

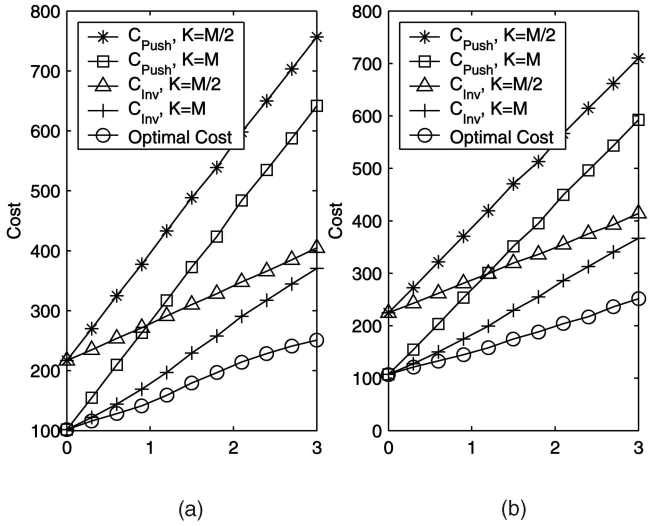

Fig. 19. Optimal cost, costs of Push and Invalidation versus U, with different update ratios. (a) Cost versus U (Uniform-SOFU). (b) Cost versus U (Exponential-SOFU).

Appendix A
Derivations of (1)-(2)

$$
\begin{aligned}
&\beta(t)\\
&= \Pr\big(\tau_3^* < \tau_2^*, \tau_0 > \tau_3^* | \tau_0 = t\big)\\
&\quad + \Pr\big(\tau_3^* > \tau_2^*, \tau_0 > \tau_3^*, L \le T | \tau_0 = t\big)\\
&= \int_{\tau_3^*=0}^{t} \int_{\tau_2^*=\tau_3^*}^{\infty} \mu e^{-\mu\tau_3^*} r_{\tau_2^*}(\tau_2^*) d\tau_2^* d\tau_3^*\\
&\quad + \Pr(L \le T) \int_{\tau_3^*=0}^{t} \int_{\tau_2^*=0}^{\tau_3^*} \mu e^{-\mu\tau_3^*} r_{\tau_2^*}(\tau_2^*) d\tau_2^* d\tau_3^*\\
&= \int_{\tau_3^*=0}^{t} \mu e^{-\mu\tau_3^*} \big[1 - R(\tau_3^*)\big] d\tau_3^*\\
&+ \Pr(L \le T) \int_{\tau_3^*=0}^{t} \int_{\tau_2^*=0}^{\tau_3^*} \mu e^{-\mu\tau_3^*} r_{\tau_2^*}(\tau_2^*) d\tau_2^* d\tau_3^*\\
&= 1 - e^{-\mu t} - \int_{\tau_3^*=0}^{t} \mu e^{-\mu\tau_3^*} R(\tau_3^*) d\tau_3^*\\
&+ \Pr(L \le T) \int_{\tau_3^*=0}^{t} \mu e^{-\mu\tau_3^*} R(\tau_3^*) d\tau_3^*\\
&= 1 - e^{-\mu t} - \Pr(L > T) \int_{\tau_3^*=0}^{t} \mu e^{-\mu\tau_3^*} R(\tau_3^*) d\tau_3^*.
\end{aligned}
$$

$$
\begin{aligned}
&P_{Adaptive\_CB}\\
&= \int_{\tau_0=0}^{\infty} \int_{\tau_1^*=\tau_0}^{\infty} \sum_{k=0}^{K-1} \binom{N-1}{k} [\beta(\tau_0)]^k [1-\beta(\tau_0)]^{N-k-1}\\
&\mu e^{-\mu\tau_0} r(\tau_1^*) d\tau_1^* d\tau_0\\
&+ \Pr(L \le T) \int_{\tau_0=0}^{\infty} \int_{\tau_1^*=0}^{\tau_0} \sum_{k=0}^{K-1} \binom{N-1}{k} [\beta(\tau_0)]^k\\
&[1-\beta(\tau_0)]^{N-k-1} \mu e^{-\mu\tau_0} r(\tau_1^*) d\tau_1^* d\tau_0\\
&= \int_{\tau_0=0}^{\infty} \sum_{k=0}^{K-1} \binom{N-1}{k} [\beta(\tau_0)]^k [1-\beta(\tau_0)]^{N-k-1}\\
&\mu e^{-\mu\tau_0} [1 - R(\tau_0)] d\tau_0\\
&+ \Pr(L \le T) \int_{\tau_0=0}^{\infty} \sum_{k=0}^{K-1} \binom{N-1}{k} [\beta(\tau_0)]^k\\
&[1-\beta(\tau_0)]^{N-k-1} \mu e^{-\mu\tau_0} R(\tau_0) d\tau_0\\
&= \int_{\tau_0=0}^{\infty} \sum_{k=0}^{K-1} \binom{N-1}{k} [\beta(\tau_0)]^k [1-\beta(\tau_0)]^{N-k-1} \mu e^{-\mu\tau_0}\\
&[1 - \Pr(L > T) R(\tau_0)] d\tau_0\\
&= \sum_{k=0}^{K-1} \binom{N-1}{k} \int_{\tau_0=0}^{\infty} [\beta(\tau_0)]^k [1-\beta(\tau_0)]^{N-k-1}\\
&\mu e^{-\mu\tau_0} [1 - \Pr(L > T) R(\tau_0)] d\tau_0.
\end{aligned}
$$

APPENDIX B
DERIVATIONS OF (11) AND (14)

Since

$$
\begin{aligned}
&\frac{d[y^{m+1}(1-y)^n]}{dy} = (m+1) y^m (1-y)^n - n y^{m+1} (1-y)^{n-1}\\
&\rightarrow y^m (1-y)^n = \frac{d[y^{m+1}(1-y)^n]}{(m+1)dy} + \frac{n y^{m+1}(1-y)^{n-1}}{(m+1)},
\end{aligned}
$$

we have

$$
\begin{aligned}
&y^m (1-y)^n =\\
&\frac{d[y^{m+1}(1-y)^n]}{(m+1)dy} + \frac{n y^{m+1}(1-y)^{n-1}}{(m+1)}\\
&= \frac{d[y^{m+1}(1-y)^n]}{(m+1)dy} + \frac{nd\Big[y^{m+2}(1-y)^{n-1}\Big]}{(m+1)(m+2)dy}\\
&+ \frac{n(n-1) y^{m+2}(1-y)^{n-2}}{(m+1)(m+2)} \qquad \text{(A.1)}\\
&= \cdots\\
&= \frac{d[y^{m+1}(1-y)^n]}{(m+1)dy} + \frac{nd\Big[y^{m+2}(1-y)^{n-1}\Big]}{(m+1)(m+2)dy} + \ldots\\
&+ \frac{n!m!d[y^{m+n+1}]}{(m+n+1)!dy}.
\end{aligned}
$$

$$
\begin{aligned}
&P_{Push}\\
&= \sum_{k=0}^{K-1} \binom{N-1}{k} \int_{\tau_0=0}^{\infty} [\beta_{Push}(\tau_0)]^k [1-\beta_{Push}(\tau_0)]^{N-k-1} \mu e^{-\mu\tau_0} d\tau_0\\
&= \sum_{k=0}^{K-1} \binom{N-1}{k} \int_{\tau_0=0}^{\infty} (1-e^{-\mu\tau_0})^k (e^{-\mu\tau_0})^{N-k-1} \mu e^{-\mu\tau_0} d\tau_0\\
&= \sum_{k=0}^{K-1} \binom{N-1}{k} \int_{x=0}^{1} (1-x)^k (x)^{N-k-1} dx, \text{ let x} = e^{-\mu\tau_0} \rightarrow d\tau_0\\
&= -\frac{1}{\mu x} dx\\
&= \sum_{k=0}^{K-1} \binom{N-1}{k} \int_{x=0}^{1} \Bigg\{ \frac{d\Big[x^{k+1}(1-x)^{N-k-1}\Big]}{(k+1)dx}\\
&\quad + \frac{(N-k-1) d\Big[x^{k+2}(1-x)^{N-k-2}\Big]}{(k+1)(k+2)dx} + \cdots\\
&\quad + \frac{k!(N-k-1)! d\big[x^N\big]}{N! dx} dx \Bigg\}, \text{ by (A.1)}\\
&= \sum_{k=0}^{K-1} \frac{(N-1)!}{k!(N-k-1)!} \frac{k!(N-k-1)!}{N!}\\
&= \frac{K}{N}.
\end{aligned}
$$

$$
\begin{aligned}
&P_{Inv}\\
&=\sum_{k=0}^{K-1}\binom{N-1}{k}\int_{\tau_0=0}^{\infty}[\beta_{Inv}(\tau_0)]^k[1-\beta_{Inv}(\tau_0)]^{N-k-1}\mu e^{-(\mu+\lambda)\tau_0}d\tau_0\\
&=\sum_{k=0}^{K-1}\binom{N-1}{k}\int_{\tau_0=0}^{\infty}\left[\frac{\mu}{\mu+\lambda}\left(1-e^{-(\mu+\lambda)\tau_0}\right)\right]^k\\
&\quad\left[1-\frac{\mu}{\mu+\lambda}\left(1-e^{-(\mu+\lambda)\tau_0}\right)\right]^{N-k-1}\mu e^{-(\mu+\lambda)\tau_0}d\tau_0\\
&=\sum_{k=0}^{K-1}\binom{N-1}{k}\int_{x=0}^{1}\left[\frac{\mu}{\mu+\lambda}(1-x)\right]^k\left[1-\frac{\mu}{\mu+\lambda}(1-x)\right]^{N-k-1}\\
&\quad\frac{\mu}{\mu+\lambda}dx,\ \text{Let } x=e^{-(\mu+\lambda)\tau_0}\rightarrow d\tau_0=\frac{-dx}{(\mu+\lambda)x}\\
&=\sum_{k=0}^{K-1}\binom{N-1}{k}\int_{y=0}^{\frac{\mu}{\mu+\lambda}}y^k(1-y)^{N-k-1}dy,\\
&\quad\text{Let } y=\frac{\mu}{\mu+\lambda}(1-x)\rightarrow dx=-\frac{\mu+\lambda}{\mu}dy\\
&=\sum_{k=0}^{K-1}\binom{N-1}{k}\int_{y=0}^{\frac{\mu}{\mu+\lambda}}\frac{d\left[y^{k+1}(1-y)^{N-k-1}\right]}{(k+1)dy}+\\
&\quad\frac{(N-k-1)d\left[y^{k+2}(1-y)^{N-k-2}\right]}{(k+1)(k+2)dy}+\cdots\\
&\quad+\frac{(N-k-1)!k!d\left[y^N\right]}{N!dy}dy,\ \text{by (A.1)}\\
&=\sum_{k=0}^{K-1}\frac{(N-1)!}{k!(N-k-1)!}\left\{\frac{y^{k+1}(1-y)^{N-k-1}}{(k+1)}+\right.\\
&\left.\frac{(N-k-1)y^{k+2}(1-y)^{N-k-2}}{(k+1)(k+2)}+\ldots+\frac{(N-k-1)!k!y^N}{N!}\right\}\Bigg|_{y=\frac{\mu}{\mu+\lambda}}\\
&=\sum_{k=0}^{K-1}\left\{\frac{(N-1)!y^{k+1}(1-y)^{N-k-1}}{k!(N-k-1)!(k+1)}+\right.\\
&\quad\left.\frac{(N-1)!y^{k+2}(1-y)^{N-k-2}}{(k+1)!(N-k-2)!(k+2)}+\ldots+\frac{(N-1)!y^N}{(N-1)!N}\right\}\Bigg|_{y=\frac{\mu}{\mu+\lambda}}\\
&=\sum_{k=0}^{K-1}\sum_{h=1}^{N-k}\binom{N-1}{k+h-1}\left\{\frac{(1-y)^{N-k-h}y^{k+h}}{(k+h)}\right\}\Bigg|_{y=\frac{\mu}{\mu+\lambda}}\\
&=\sum_{k=0}^{K-1}\sum_{h=1}^{N-k}\binom{N-1}{k+h-1}\left[\frac{\lambda^{N-k-h}\mu^{k+h}}{(k+h)(\mu+\lambda)^N}\right]\\
&=\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N-1}{n-1}\left[\frac{\lambda^{N-n}\mu^n}{n(\mu+\lambda)^N}\right],\ \text{let } n=h+k\\
&=\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\frac{(N-1)!}{(n-1)!(N-n)!}\left[\frac{\lambda^{N-n}\mu^n}{n(\mu+\lambda)^N}\right]\\
&=\frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}\lambda^{N-n}\mu^n}{N(\mu+\lambda)^N}\\
&=\frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N}.
\end{aligned}
$$

APPENDIX C
PROOF OF LEMMA 1

$$
\begin{aligned}
&\lim_{U\rightarrow\infty}\frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N}\\
&=\lim_{U\rightarrow\infty}\frac{\sum_{k=0}^{K-1}\binom{N}{k+1}U^{N-(k+1)}}{NU^N}=\lim_{U\rightarrow\infty}\frac{\binom{N}{1}U^{N-1}}{NU^N}=0.
\end{aligned}
$$

$$
\begin{aligned}
\lim_{U\rightarrow\infty}T_{Optimal}&=\lim_{U\rightarrow\infty}L_{Inv}\left\{U+\tfrac{K}{N}\right\}\Big/\left(U-\tfrac{K}{N}\right)\\
&=L_{Inv.}
\end{aligned}
$$

$$T_{Optimal}\geq L_{Inv}.$$

$$
\begin{aligned}
&\Leftrightarrow U-\frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N}+\frac{K}{N}\\
&\quad\geq U+\frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{N(1+U)^N}-\frac{K}{N}.
\end{aligned}
$$

$$\Leftrightarrow\frac{\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}}{(1+U)^N}\leq K.$$

$$\Leftrightarrow\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}\leq K\sum_{n=0}^{N}\binom{N}{n}U^{N-n}.$$

$$\Leftrightarrow\sum_{k=0}^{K-1}\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}\leq\sum_{k=0}^{K-1}\sum_{n=0}^{N}\binom{N}{n}U^{N-n}.$$

$$\Leftrightarrow\sum_{k=0}^{K-1}\left[\sum_{n=0}^{N}\binom{N}{n}U^{N-n}-\sum_{n=k+1}^{N}\binom{N}{n}U^{N-n}\right]\geq 0.$$

It is obvious that the last inequality is true. □

APPENDIX D
DERIVATION OF (32)

$$
\begin{aligned}
&\frac{\partial C_{Adpative-CB}}{\partial T} \\
&= \left[-\frac{\partial A(T)}{\partial T} - \gamma e^{-\gamma T} B(T) + e^{-\gamma T}\frac{\partial B(T)}{\partial T}\right]\left(L_{\mathrm{Req}} + \tfrac{1}{\gamma}\right) \\
&\quad + U\left[-\gamma L_{Inv} e^{-\gamma T} + \gamma e^{-\gamma T}\left(\tfrac{1}{\gamma} - T e^{-\gamma T} - \tfrac{1}{\gamma} e^{-\gamma T}\right)\right. \\
&\quad \left. + \left(-e^{-\gamma T} + \gamma T e^{-\gamma T} + e^{-\gamma T}\right)\left(1 - e^{-\gamma T}\right)\right] \\
&= \left[-\frac{\partial A(T)}{\partial T} - \gamma e^{-\gamma T} B(T) + e^{-\gamma T}\frac{\partial B(T)}{\partial T}\right]\left(L_{\mathrm{Req}} + \tfrac{1}{\gamma}\right) \\
&\quad + U\left[-\gamma L_{Inv} e^{-\gamma T} + \tfrac{1}{\gamma}\gamma e^{-\gamma T} - T e^{-\gamma T}\gamma e^{-\gamma T}\right. \\
&\quad \left. - \tfrac{1}{\gamma} e^{-\gamma T}\gamma e^{-\gamma T} + \gamma T e^{-\gamma T} - \gamma T e^{-\gamma T} e^{-\gamma T}\right] \\
&= \left[-\frac{\partial A(T)}{\partial T} - \gamma e^{-\gamma T} B(T) + e^{-\gamma T}\frac{\partial B(T)}{\partial T}\right]\left(L_{\mathrm{Req}} + \tfrac{1}{\gamma}\right) \\
&\quad + U\left[-\gamma L_{Inv} e^{-\gamma T} + e^{-\gamma T} - T e^{-2\gamma T} - \tfrac{1}{\gamma} e^{-2\gamma T}\right. \\
&\quad \left. + \gamma T e^{-\gamma T} - \gamma T e^{-2\gamma T}\right].
\end{aligned}
$$

ACKNOWLEDGMENTS

The authors would like to express their great thanks to anonymous reviewers for their valuable comments, which have significantly improved the quality of this paper.

REFERENCES

[1] WAP Forum, "Wireless Application Protocol Architecture Specification," technical report, WAP Forum, 1998.
[2] WAP Forum, "Wireless Application Protocol White Paper," technical report, WAP Forum, 1999.
[3] WAP Forum, "Wireless Application Protocol V1.1 to V1.2," technical report, WAP Forum, 1999.
[4] C.H. Rao, D.-F. Chang, and Y.-B. Lin, "iSMS: An Integration Platform for Short Message Service and IP Networks," *IEEE Network,* vol. 15, no. 2, pp. 48-55, 2001.
[5] 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. 1247-1258, Nov. 2003.
[6] WAP Forum, "Wireless Application Protocol Cache Model Specification," technical report, WAP Forum, 1998.
[7] 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, July/Aug. 1999.
[8] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *Proc. ACM Mobicom,* 2000.
[9] G. Cao, "Proactive Power-Aware Cache Management for Mobile Computing Systems," *IEEE Trans. Computers,* vol. 51, no. 6, pp. 608-621, June 2002.
[10] L. Breslau, P. Cao, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM '99,* pp. 126-134, Mar. 1999.
[11] S. Khanna and V. Liberatore, "On Broadcast Disk Paging," *SIAM J. Computing,* vol. 29, no. 5, pp. 1683-1702, 2000.
[12] 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/Aug. 1999.
[13] S.M. Ross, *Stochastic Processes.* New York: Wiley, 1996.
[14] Y.-B. Lin, "Reducing Location Update Cost in a PCS Network," *IEEE/ACM Trans. Networking,* vol. 5, no. 1, pp. 25-33, Feb. 1997.
[15] Y. Xiao, "Optimal Location Management for Two-Tier PCS Networks," *Computer Comm.,* vol. 26, no. 10, pp. 1047-1055, June 2003.
[16] L. Berslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM,* vol. 1, pp. 126-134, Mar. 1999.
[17] G.K. Zipf, "Relative Frequency as a Determinant of Phonetic Change," reprinted from the *Harvard Studies in Classical Philology,* vol. XL, 1929.
[18] X. Du, "QoS Routing Based on Multi-Class Nodes for Mobile Ad Hoc Networks," *Elsevier J. Ad Hoc Networks,* vols. 2/3, pp. 241-254, July 2004.
[19] H. Chen and Y. Xiao, "Cache Access and Replacement for Future Wireless Internet," *IEEE Comm. Magazine,* pp. 113-123, May 2006.

Yang Xiao worked at Micro Linear as a MAC architect involving work on the IEEE 802.11 standard enhancement before he joined academia in 2002 as a faculty member. Dr. Xiao is now the director of the W4-Net Lab and was previously with the Center for Information Assurance. He is currently with the Department of Computer Science at The University of Alabama. He was a voting member of the IEEE 802.11 Working Group from 2001 to 2004 and is a senior member of the IEEE. He currently serves as editor-in-chief for the *International Journal of Security and Networks* (IJSN) and for the *International Journal of Sensor Networks* (IJSNet). He serves as an associate editor or on editorial boards for the following refereed journals: the *International Journal of Communication Systems* (Wiley), *Wireless Communications and Mobile Computing* (WCMC), the *EURASIP Journal on Wireless Communications and Networking* (WCN), and the *International Journal of Wireless and Mobile* Computing (IJWMC). He has served as a guest editor for seven journal special issues and serves as a referee/reviewer for many funding agencies, as well as a panelist for the US National Science Foundation and a member of the Canada Foundation for Innovation (CFI)'s Telecommunications expert committee. He serves as TPC for more than 70 conferences, such as INFOCOM, ICDCS, ICC, GLOBECOM, WCNC, etc. His research areas are wireless networks, mobile computing, and network security. He has published more than 140 papers in major journals and refereed conference proceedings related to these research areas.

Hui Chen received the MS degree in computer science from the University of Memphis, Memphis, Tennessee, in 2003. He is currently a PhD candidate in computer science at the University of Memphis. His current research interests include the design and analysis of personal communication service networks, wireless LANs, and mobile distributed computer systems. He is a student member of the IEEE.