Wireless Networks 11, 243–254, 2005

Strongly Consistent Access Algorithms for Wireless Data Networks

YUGUANG FANG*
Department of Electrical and Computer Engineering, University of Florida, 435 Engineering Building, P.O. Box 116130, Gainesville, FL 32611

YI-BING LIN
Dept. Comp. Sci. & Info. Management, Providence University, TaiChung, Taiwan

Abstract. In wireless data networks such as the WAP systems, the cached data may be time-sensitive and strong consistency must be maintained (i.e., the data presented to the user at the WAP handset must be the same as that in the origin server). In this paper, we study the cached data access algorithms in such systems. Two caching algorithms are investigated. In Algorithm I, *Pull-Each-Read*, whenever a data access occurs, the client always asks the server whether the cached entry in the client is valid or not. In Algorithm II, *Callback*, the server always invalidates the cached entry in the client whenever an update occurs. Analytic models are proposed to evaluate the performance of these algorithms. Our studies show that Algorithm II outperforms Algorithm I if the data access rate is high and the access pattern is irregular. We also design an adaptive mechanism to effectively switch between the two algorithms to take advantages of both algorithms. We also apply the single-level cached data access algorithms for the multi-level cache hierarchy. Our study indicates that with appropriate arrangement, strongly consistent cached data access for wireless Internet (such as WAP) can be efficiently supported.

Keywords: cache access, strong consistency, wireless application protocol (WAP), wireless internet

1. Introduction.

The increasing Internet and multimedia applications are a major factor that drives the new generation mobile network technology. It was indicated that more than 20% of the adult population of the US are interested in wireless Internet access. As the advanced wireless infrastructure becomes available and the inexpensive wireless handheld devices (e.g., wireless personal data assistant and wireless smart phones) become popular, the subscribers can enjoy instant wireless Internet access. The services include sales force automation, dispatch, instant content access, banking, e-commerce, and so on.

In a mobile network, however, wireless data are delivered with several constraints. In order to be portable, size and weight establish limits for wireless handheld devices, which result in a restricted user interface (small displays and keypads), less powerful CPU, low transmission power and reduced memory capacity. Also, compared with its wireline counterpart, the wireless network has limited bandwidth, limited transmission power, longer latency, and a lower degree of reliability. These limitations should be carefully addressed so that the wireless handheld devices can access Internet applications that are typically designed for desktop computers.

In June 1997, Ericsson, Motorola, Nokia and Phone.com founded the *Wireless Application Protocol* (WAP) Forum. The WAP forum has drafted a set of global wireless protocol specifications for many wireless networks, which has been contributed to various industry groups and standards bodies. Most handset manufacturers have committed to WAP-enabled devices, and many mobile operators have joined the WAP Forum. WAP has attracted significant attention for the following reason. Until early 2000, wireless data services have not been as successful as mobile network operators expected. By providing a better environment to integrate Internet, WAP is anticipated to significantly improve the wireless data market.

The WAP architecture [14,16,17] is illustrated in figure 1(a). In this architecture, the WAP Gateway interworks the wireless network with the IP network, which allows a WAP handset to obtain data from an origin server. To converge wireless data and the Internet, WAP integrates a light-weight web browser into handheld devices with limited computing and memory capacities. The wireless application protocols implemented in the WAP Gateway and the WAP handset enable a mobile user to access Internet web applications through a client/server model. Furthermore, WAP supports *User Agent Profile* (UAProf) [15] that allows a WAP handset to describe its capability to application servers or other network entities so that the servers/network entities can generate contents based on the handset's capability. This feature allows the applications to exploit the maximum potential of WAP handsets.

To speed up wireless data access, the WAP user agent caching model was proposed [15], which tailors the HTTP caching model to support WAP handsets with limited functions. Figure 1(b) shows a two-level cache hierarchy in WAP environment. In this hierarchy, the WAP Gateway caches the data entries from the origin server, and the WAP handset caches the data entries from the WAP Gateway.

For cached items that will not be changed during user retrievals, they can be efficiently accessed by the WAP handsets without revalidation. On the other hand, a time-sensitive cached item is set to "must-revalidate". If this cached item is

* Corresponding author.
E-mail: fang@ece.ufl.edu

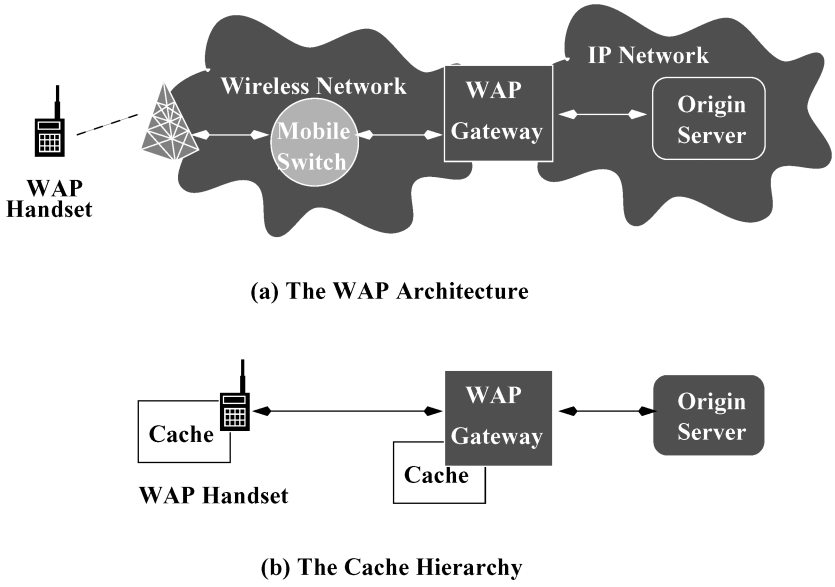


Figure 1. The WAP architecture and a two-level caching model.

stale when the user tries to go back in the history, then the user agent revalidates this cached item. In general, navigation and processing within a single cached item does not require revalidation except for the first fetch.

The HTTP caching model is sensitive to time synchronization. Since WAP follows this model, a reliable time-of-day clock should be maintained in the WAP Gateway. If a WAP user agent does not have access to a time-of-day clock, it should exchange the time-of-day request and response message with the WAP Gateway and synchronize with the clock value returned from the WAP Gateway. Another important issue for caching is security. The private information in the user agent cache is protected from unintended or malicious access. WAP Gateways implementing a caching function must obey all security-related considerations defined in HTTP.

Besides caching, complementary techniques such as *prefetching* and *push* can also be used to speed up the web access. Prefetching is based on the fact that after retrieving a page from the origin server, the WAP user may spend some time viewing the page. During this period the WAP handset as well as the air link of the wireless network are idle, providing time which can be used to prefetch the next page. Thus when the user proceeds to retrieve the information, the page is available immediately. In this way, transmission delay can be reduced. However, if the prefetched data is not used by the user, the network resources used by prefetching are wasted. Of course, if the prefetching is based on the best-of-effort, that is, prefetch only whenever the resource is idle, then such network resource wastage may not be a serious problem.

In the push mechanism, a trusted application server can send information directly to the application environment for processing. Push allows applications to alert the user when time-sensitive information changes. Application-generated events, such as telephony applications and emergency services, can benefit from using push technology. To fully utilize the push mechanism, users are allowed to register their interests (e.g., when and how often a data object should be pushed) to the server. The server then pushes the data objects based on the users' interests.

This paper studies cached data access algorithms in wireless environments such as WAP. We assume that the cached data are time-sensitive and strong consistency must be maintained (i.e., the data presented to the user at the WAP handset must be the same as that in the origin server). We first describe two strongly consistent cached data access algorithms. Then we propose an analytic model to evaluate the communication costs of these algorithms. An adaptive mechanism is designed to effectively switch between the two algorithms to reduce the communication costs. Then we apply the single-level cached data access algorithms for the multi-level cache hierarchy. Our study indicates that with appropriate arrangement, strongly consistent cached data access for wireless Internet (such as WAP) can be efficiently supported. A WAP prototype (with push mechanism) has been developed in National Chiao Tung University. One of our future research directions is to implement the adaptive mechanism for the strongly consistent cached data access algorithms in our WAP prototype.

2. Strongly consistent cached data access algorithms

This section describes strongly consistent cached data access algorithms for wireless WAP systems. For the discussion purpose, we define the term "server" as the node that supplies data, and the term "client" as the node that caches the data entries received from the server. In figure 1(b), the origin server and the WAP Gateway are a server-client pair, and the WAP Gateway and the WAP handset are another server-client pair.

Several strongly and weakly consistent cached data access algorithms were given in [18] for wireline computer networks. In this paper, we investigate two strongly consistent algorithms for wireless applications: *poll-each-read* and *callback*. In

poll-each-read [5], the client always asks the server if the cached entry is valid. If so, the server responds affirmatively. Otherwise, the server sends the current data entry to the client. There are two approaches to implement poll-each-read. In the first approach, timestamps are used to indicate if a cached entry is up to date. The timestamp of a cached entry in the client indicates when the entry was received by the client. The timestamp of the entry in the server indicates the latest time when the entry was updated. Thus, by comparing the timestamps of the entry in the client and the server, the server determines if the cached entry in the client is out of date. The timestamp of a cached entry can be maintained by the client or by the server. If the client is a WAP handset, the cached capacity is limited. In this case, the information should be maintained by the server. This implies that the server should have a caching timestamp table for each client. In WAP, this table can be implemented in UAProf, a User Agent Profile discussed in [9].

Instead of using timestamps, we propose the following implementation for poll-each-read using validation bits. In this approach, the server maintains a validation bit c_v for each of the clients who have the cached entry. The algorithm works as follows:

Algorithm I. Poll-Each-Read

I.1. Entry Update (Server). When a data entry is updated, for every client that has the cached entry, the server sets c_v to 0. Note that "0" implies that the entry is invalidated.

I.2. Entry Access (Client). To access a data entry, a client sends an entry access message to the server. The message contains an access type bit c_a. If the client does not have a cached entry (either the entry is first accessed or was replaced), then c_a is 1. In this case, the entry in the server should be sent to the client. If the client has the cached entry, then c_a is set to 0. In this case, the cached entry should be validated by the server.

I.3. Entry Access (Server). The server receives an entry access message from a client. Let c_v be the validation bit for that client.

I.3.1. If the client does not have the cached entry (i.e., $c_a = 1$), the server sends the entry to the client, and c_v is set to 1.

I.3.2. If $c_a = 0$ and $c_v = 0$, then the server sends the data entry to the client. The bit c_v is set to 1.

I.3.3. If $c_a = 0$ and $c_v = 1$ then the server returns validation affirmation to the client.

Figure 2 shows the messages exchanged between the server and the client for a update and access sequence of a data entry with poll-each-read. In this example, accesses occur at times t_0, t_1 and t_3. An update occurs at time t_2. The first access to the data entry occurs at time t_0. Since the client does not have a cache copy, it sends an entry access message with $c_a = 1$ to the server (Step I.2). The server sends the data entry to the client and sets the validation bit c_v to 1 (Step I.3.1). The second access request occurs at time t_1. Since the client already has a cache copy, it sends an entry access message with $c_a = 0$. The server finds that $c_v = 1$ and the cached entry in the client is valid. The server sends an affirmation to the client (Step I.3.3). The client then uses the cached entry. At time t_2, an update to the data entry occurs. The c_v bit is set to 0 (Step I.1). When another access occurs at time t_3, the data entry is sent from the server to the client (Step I.3.2) because the cached entry in the client is invalid.

In the *callback* approach [5,12], when modifying a data entry, the server either invalidates the cached entry in the clients or send the current data entry to the clients. Figure 3(a) gives a scenario for entry update and access with callback, where the current data entry is always sent to the client when it is updated at the server. In this example, the first access occurs at time t_0. Since the client does not have a cache copy, the data is obtained from the server. At time t_1, the second access occurs and the client utilizes the cached entry. Then four consecutive updates occur at times t_2, t_3, t_4, and t_7, respectively. The data entry is sent from the server to the client at each update. At times t_5, t_6, and t_8, the accesses occur and the cached entry is utilized. In the above scenario, all data accesses are performed locally at the client. Thus, the data access can be done quickly. However, the requirement of the entry sending per update may unnecessarily consume transmission bandwidth. In figure 3(a), the data sendings at time t_2 and t_3 are not necessary because the data entry is not accessed before time t_5. These unnecessary data sending can be avoided if the server invalidates the client copy when an update occurs. We propose the following modified algorithm where the server maintains a validation bit c_v for each of the clients that have the cached entry.

Algorithm II. Callback

II.1. Entry Update (Server). When an update occurs, for every client that has the cached entry, if $c_v = 1$, the server sends an invalidation message to the client. Then the server sets c_v to 0.

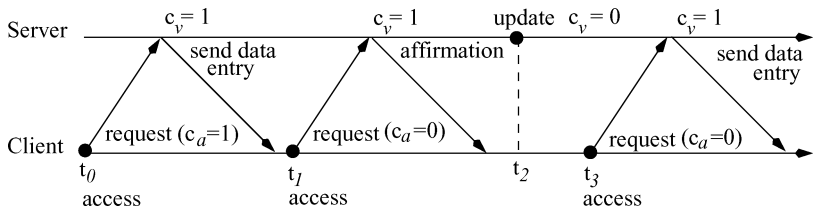
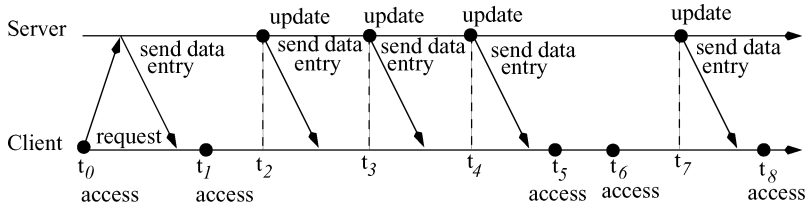
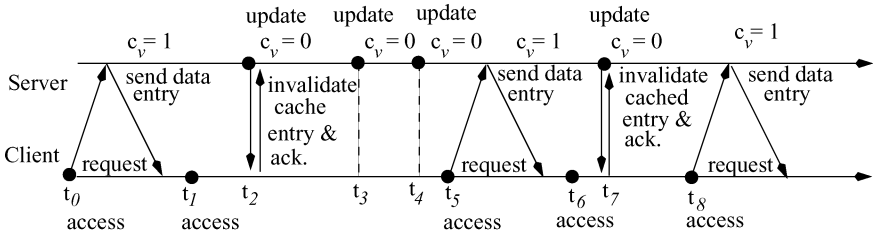


Figure 2. Data entry access and update in poll-per-read.



(a) Sending Data Entry for Every Update



(b) Invalidating Cache Entry for Every Update

Figure 3. Data entry access and update in callback.

II.2. Entry Update (Client). When the client receives the invalidation message, the cached entry is invalidated and the storage can be reclaimed to cache another data entry. The client sends an acknowledgment message to the server.

II.3. Entry Access (Client). If the cached entry exists, then the client uses the cached entry. Otherwise, the client sends an entry access message to the server. Eventually, the client will receive the data entry from the server.

II.4. Entry Access (Server). When the server receives an entry access message from a client, it sends the data entry to the client. Let c_v be the validation bit for that client. The server sets c_v to 1.

From the above description, we observe the following fact:

Fact 1. In Algorithm II, a data entry is in the cache if the cached entry is valid.

Algorithm II attempts to avoid unnecessary updates described in figure 3(a). Figure 3(b) shows the communication between the server and the client for the update and access sequence given in figure 3(a). In this scenario, the data entry is sent to the client three times (comparing to five times in figure 3(a)). The disadvantage of Algorithm II is that the access delay at time t_8 is longer than that in figure 3(a). Compared with Algorithm I, Algorithm II does not require the client to check with the server for every data access. On the other hand, for data invalidation in Algorithm II, the server needs to invalidate all clients who have the cached entry. In the subsequent sections, we analyze the performance of Algorithms I and II.

3. Analytical models for single-level cache

To our understanding, most performance studies for web caching were conducted by tracing specific web applications. For example, the work in [19] utilized log files from Boston University. While the trace driven approach can study the behavior of specific cases, the results may not be generalized. In this and the subsequent sections, we propose analytical model to analyze Pull-Each-Read and Callback. With the Gamma distributions for the access and the update patterns, general conclusions (such as the impact of the variance of the access/update patterns) can be drawn in our study.

Let α be the probability that there are at least one update between two data accesses. Let x be the cost for data entry access message or validation/invalidation message, and y be the cost for delivering a data entry from the server to the client. Both x and y can be measured by bytes. Note that the costs x and y may not be different if the radio characteristics of uplink and downlink transmissions are considered. For simplicity, we assume that costs for the data entry access message, validation message, and invalidation message are all the same. It is clear that our results can be easily extended for the case that they are not the same. In Algorithm I, for every data access, the client sends an entry access message to the server at Step I.2 (with cost x). With probability α the entry is sent back to the client at Step I.3.2 (with cost y) and with probability $1 - \alpha$ a validation affirmation message is sent from the server to the client at Step I.3.3 (with cost x). Thus for Algorithm I, the communication cost C_I per data access is

$$C_I = x + \alpha y + (1 - \alpha)x = \alpha(y - x) + 2x \qquad (1)$$

Consider data access in Algorithm II. If the cached entry is valid (with probability $1 - \alpha$), the client utilizes the cached entry without any communication between the client and the server. If the cached entry is obsolete, then an invalidation message pair is exchanged by the client and the server at Steps II.1 and II.2 (with cost $2x$). To access a data entry, the client sends an entry access message to the server at Step II.3 (with cost x) and the server returns the valid entry to the client at Step II.4

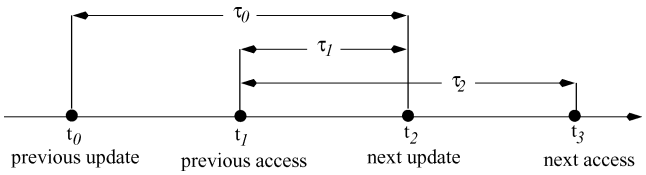


Figure 4. Timing diagram for deriving α.

(with cost y). Thus the communication cost C_{II} per data access for Algorithm II is

$$C_{II} = \alpha(2x + x + y) = \alpha(3x + y) \tag{2}$$

Remark. We may apply some cost factors to indicate the cost difference of messages flowing in different directions, which may reflect different transmission limitations. For example, between the gateway and the WAP handset, the transmission power for WAP handset may be critical due to battery-powered nature, we may use high cost coefficient to the messages from the WAP handset than the ones from the WAP gateway. However, we will not explicitly express this cost factor in this paper for simplicity of presentation, most results can be easily modified if we do.

In (1) and (2), the probability α plays an important role, which can be derived as follows. Consider the timing diagram in figure 4. Suppose that an access arrives at time t_1. The last update before t_1 occurs at time t_0. The next update after t_1 occurs at time t_2 and the next access occurs at time t_3. The inter arrival time of update is $\tau_0 = t_2 - t_0$, and the inter arrival time of access is $\tau_2 = t_3 - t_1$. Let $\tau_1 = t_2 - t_1$ be the period between an access and the next update. Assume that both τ_0 and τ_2 are random variables. Let f_u be the density function of τ_0 with mean $1/\lambda$ and f_a be the density function of τ_2 with mean $1/\mu$, both of which have Laplace transforms $f_u^*(s)$ and $f_a^*(t)$, respectively. Let $r_u(t)$ and $r_u^*(s)$ be the probability density function and Laplace transform for τ_1. Let random variable K be the number of updates occurring between two accesses. Then

$$\alpha = \Pr[K > 0] = \Pr[\tau_1 < \tau_2]$$

Following similar argument used in [4], we obtain

$$\begin{aligned}
\alpha &= \Pr(\tau_1 \le \tau_2) \\
&= \int_0^\infty \Pr(\tau_1 \le t) f_a(t)\, dt \\
&= \int_0^\infty \left(\frac{1}{2\pi j}\right) \left\{ \int_{\sigma - j\infty}^{\sigma + j\infty} \left[\frac{r_u^*(s)}{s}\right] e^{st} ds \right\} f_a(t)\, dt \\
&= \left(\frac{1}{2\pi j}\right) \int_{\sigma - j\infty}^{\sigma + j\infty} \left[\frac{r_u^*(s)}{s}\right] \left[\int_0^\infty f_a(t) e^{st}\, dt \right] ds \\
&= \left(\frac{1}{2\pi j}\right) \int_{\sigma - j\infty}^{\sigma + j\infty} \left[\frac{r_u^*(s)}{s}\right] f_a^*(-s) ds
\end{aligned}$$

where σ is sufficiently small positive number. Let σ_a denote the set of poles of $f_a^*(-a)$ in the strict right complex plane, applying the Residue Theorem, we obtain

$$\begin{aligned}
\alpha &= -\sum_{p \in \sigma_a} Res_{s=p} \left[\frac{r_u^*(s)}{s}\right] f_a^*(-s) \\
&= -\sum_{p \in \sigma_a} Res_{s=p} \left[\frac{\lambda(1 - f_u^*(s))}{s^2}\right] f_a^*(-s)
\end{aligned} \tag{3}$$

Similarly, we can obtain

$$\begin{aligned}
\alpha &= 1 + \sum_{p \in \sigma_u} Res_{s=p} \left[\frac{f_a^*(s)}{s}\right] r_u^*(-s) \\
&= 1 - \sum_{p \in \sigma_u} Res_{s=p} \left[\frac{\lambda f_a^*(s)}{s^2}\right] (1 - f_u^*(-s)) \\
&= 1 + \sum_{p \in \sigma_u} Res_{s=p} \left[\frac{\lambda f_a^*(s)}{s^2}\right] f_u^*(-s)
\end{aligned} \tag{4}$$

where σ_u is the set of poles of $r_u^*(-u)$ and $f_u^*(-s)$ in the right half complex plane (notice that $r_u^*(-s)$ and $f_u^*(-s)$ share the same set of poles). For the demonstration purpose, we consider two cases:

Case A. When τ_0 has an exponential distribution (i.e., the arrivals of updates form a Poisson process where $f_u(\tau_0) = \lambda e^{-\lambda \tau_0}$) and τ_2 has a general distribution, from (4) we can easily obtain

$$\alpha = \Pr[K > 0] = 1 - f_a^*(\lambda) \tag{5}$$

Case B. When τ_0 has a general distribution and τ_2 has an exponential distribution where $(f_a(\tau_2) = \mu e^{-\mu \tau_2})$, from (3) we obtain

$$\alpha = \Pr[K > 0] = r_u^*(\mu) = \left(\frac{\lambda}{\mu}\right) [1 - f_u^*(\mu)] \tag{6}$$

Since the probability α will be extensively used in the subsequent development, we present another approach for its computation. This approach is based on distribution model called *phase-type distribution*. It is well-known that [8] the phase-type distributions (PH) are dense in the set of all distributions in $[0, \infty)$, i.e., any distribution of a nonnegative random variable can be approximated by Phase-type distributions. The exponential distribution, the Erlang distributions, the hyper-exponential distribution and the hyper-Erlang distribution are all special cases of PH distributions. The advantage of PH distributions is that most computations are reduced to matrix manipulations. A PH distribution is the distribution of the time till absorption into the absorbing state 0 in a finite state Markov chain with states $\{0, 1, 2, \ldots, n\}$ and with initial probability vector (ζ_0, ζ) and infinitesimal generator

$$Q = \begin{pmatrix} 0 & \mathbf{0} \\ \mathbf{t} & T \end{pmatrix}$$

where ζ is row vector of size n and T is an $n \times n$ matrix. It can be shown [8] that this distribution can be uniquely determined by the pair (ζ, T), so we say that a random variable X is $PH(\zeta, T)$ if X is PH distributed with parameter (ζ, T). In fact, a random

variable with $PH(\zeta, T)$ has the following probability density function

$$f(x) = -\zeta \exp(Tx)T\mathbf{1}_T, \quad x \geq 0$$

where $\mathbf{1}_T$ is the column vector of all 1's with the same dimension as the matrix T (i.e., if T is an $n \times n$ matrix, then $\mathbf{1}_T$ will be an n-dimensional vector). We need the following result:

Lemma 1 [1,8].

(1) Assume that $F(x)$ is the cumulative distribution function of a random variable with $PH(\zeta, T)$ with expectation $1/\mu$, then the distribution $\mu[1 - F(x)]$ is also PH distributed with $PH(\pi, T)$ where $\pi = (\zeta T^{-1}\mathbf{1}_T)^{-1}\zeta T^{-1}$.

(2) Assume that random variables X and Y are independent with $PH(\zeta, T)$ and $PH(\nu, S)$, respectively, then

$$\Pr(X \leq Y) = (\nu \otimes \zeta)(-S \oplus T)^{-1}(\mathbf{1}_S \otimes (-T\mathbf{1}_T)).$$

where $\otimes$ indicates the Kronecker product and $\oplus$ denotes the Kronecker sum. Assume that the inter-update time τ_0 is $PH(\zeta, T)$ and inter-access time τ_2 is $PH(\nu, S)$, then from Lemma 1 we conclude that τ_1 is also PH distributed with $PH(\pi, T)$ with $\pi = (\zeta T^{-1}\mathbf{1}_T)^{-1}\zeta T^{-1}$. Applying Lemma 1 again, we obtain

$$\alpha = \Pr(\tau_1 \leq \tau_2) = (\nu \otimes \pi)(-S \oplus T)^{-1}(\mathbf{1}_S \otimes (-T\mathbf{1}_T)),$$
$$\pi = (\zeta T^{-1}\mathbf{1}_T)^{-1}\zeta T^{-1}. \tag{7}$$

By directly applying PH distributions (e.g., hyper-exponential and Erlang) into (5), (6), and (7), it is apparent that these equations are the same. Equations (5) and (6) can be quickly computed when the distributions have simple form for Laplace transforms. On the other hand, (7) can be easily used for Phase-type distributions which may have complicated Laplace transforms.

Based on (5) and (6), figure 5 plots the α values for Cases A and B. In this figure, the Gamma distributions are considered for τ_2 in Case A and τ_0 in Case B. Thus,

$$f_a^*(s) = \left(\frac{1}{1 + \mu v s}\right)^{\frac{1}{\mu^2 v}} \quad \text{and} \quad f_u^*(s) = \left(\frac{1}{1 + \lambda v s}\right)^{\frac{1}{\lambda^2 v}}$$

in Cases A and B, respectively, where v is the variance of the Gamma distribution. The Gamma distribution is selected because it can be used to approximate many other distributions as well as measured data [6,7]. Gamma distributions are used in several mobile network studies [1–4]. Figure 5 indicates that both Cases A and B show the same trends for the α curves, namely, α decreases as μ/λ and v increase. In the remainder of this paper, we only consider Case A. The results for Case B are similar and are omitted.

Based on (1) and (2), figure 6 plots C_I and C_{II} against μ, where both τ_0 and τ_2 are exponentially distributed. The figure indicates that when μ is small $C_{II} > C_I$, and when μ is large, $C_I > C_{II}$. That is, Algorithm I outperforms Algorithm II when there are more updates. On the other hand, Algorithm II outperforms Algorithm I when there are more accesses.

Figure 7 shows the impact of variance v of the τ_2 (inter access time) distribution where the inter update times are exponentially distributed and $y = 10x$. We observe that when v increases, both C_I and C_{II} decrease. This phenomenon is explained as follows. Large variance implies that the data access pattern becomes irregular (i.e., there are more long and short τ_2). For those short τ_2, it is likely that no update occurs between two accesses. For those long τ_2, many updates may occur between two accesses. However, only one of them will cause the client to access the entry from the server.

We also observe that when update frequency is high (i.e., $\mu = 0.1\lambda$ in figure 7), Algorithm II may outperforms Algorithm I if v is sufficiently large. That is, Algorithm II benefits more on the increase of v than Algorithm I does.

The analysis in this section assumes that the cached entries in the client are not replaced. This assumption may hold when the client is the WAP Gateway (where memory storage is large). However, it is not true if the client is a WAP handset. In this case, (1) and (2) are re-written as

$$C_I = (1 - \gamma_I)(x + y) + \gamma_I\Big[\alpha(y - x) + 2x\Big]$$
$$C_{II} = (1 - \gamma_{II})(x + y) + \alpha\gamma_{II}(3x + y)$$

where γ_I and γ_{II} are the cache hits (the probability that the cached entry is found at an access) for Algorithms I and II,

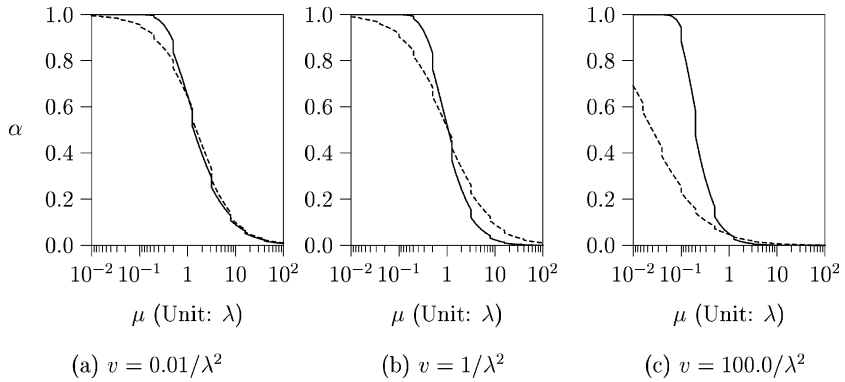


Figure 5. The α values for various entry access and update distributions (Solid curves: Case A; dashed curves: Case B).

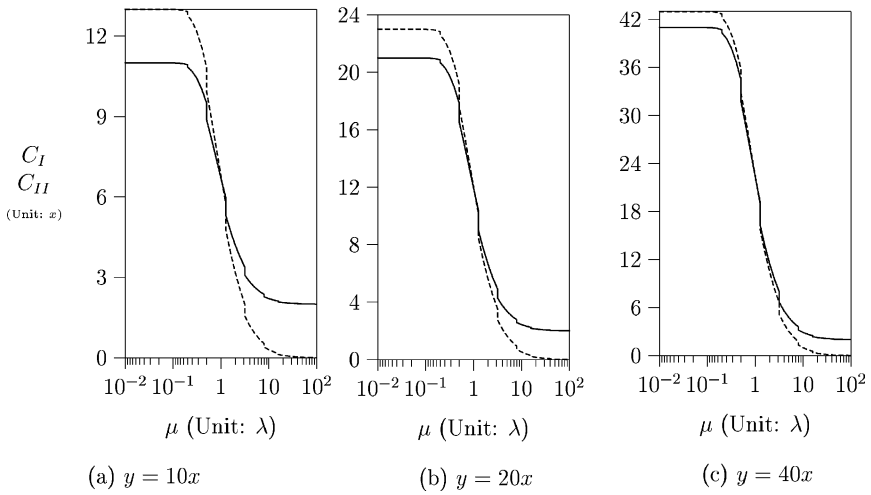


Figure 6. Impact of μ/λ on C_I and C_{II} (Solid curves: C_I; dashed curves: C_{II}).

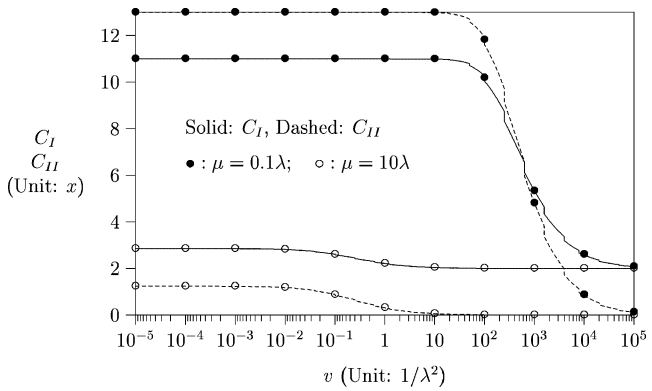


Figure 7. Impact of distribution variance v on C_I and C_{II}.

respectively. Note that when exercising the same cache replacement algorithm, the cache hits may be different for these two algorithms. As described in the next section, $\gamma_{II} \geq \gamma_I$ is expected. Due to the page limitation, the impact of the cache replacement algorithms (such as LRU) will not be elaborated in this paper, but will be addressed elsewhere [10].

4. Other performance measures for single-level cache

This section considers two performance measures other than the communication costs. We first study the number of invalidation messages sent from the server to the client in Algorithm I. Then we investigate how long the stale cached entry will occupy cache storage in Algorithm II.

4.1. Invalidation messages sent in callback

If there are K updates between two accesses, then the data entry is sent to the client K times in the traditional callback algorithm (see the example in figure 3(a)). In Algorithm II, only the first one of these K updates results in an invalidation message sent from the server to the client. Let β be the portion of updates that can be ignored in Algorithm II. Then

$$\beta = \frac{E[(K-1)^+]}{E[K]}$$

where

$$X^+ = \begin{cases} X & X \geq 0 \\ 0 & X < 0 \end{cases}$$

From the Little's law [13]

$$E[K] = \sum_{k=0}^{\infty} k \Pr[K = k] = \frac{\lambda}{\mu}$$

The expected value $E[(K-1)^+]$ is derived as follows:

$$\begin{aligned} E[(K-1)^+] &= \sum_{k=1}^{\infty} (k-1) \Pr[K = k] \\ &= E[K] - \Pr[K > 0] = E[K] - \alpha \end{aligned}$$

and

$$\beta = 1 - \left(\frac{\mu}{\lambda}\right) \alpha$$

where the computation of α was given in the previous section. From (5), when the updates form a Poisson process, we have

$$\beta = 1 - \left(\frac{\mu}{\lambda}\right) [1 - f_a^*(\lambda)] \tag{8}$$

From (6), when the accesses form a Poisson process, we have

$$\beta = 1 - \left(\frac{\lambda}{\mu}\right) \left(\frac{\mu}{\lambda}\right) [1 - f^*(\mu)] = f_u^*(\mu)$$

Figure 8 plots the β curves based on (8). We assume that τ_2 has a Gamma distribution with mean $1/\mu$ and variance v. The figure indicates that Algorithm II can ignore most updates when the data access pattern becomes irregular (i.e., when v is large). The variance v has more impact on β when $\frac{\mu}{\lambda}$ is large.

As we mentioned before, for an update in Algorithm II, the server needs to invalidate all clients who have the cached

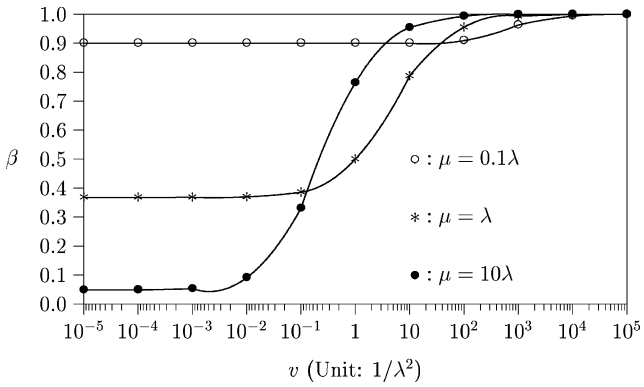


Figure 8. The portion β of invalidation messages saved in callback.

entry. For every client, the communication cost for this operation is already considered in (2). For the server, the processing cost for sending invalidation messages to multiple clients may be expensive. In WAP, the applications in origin server are typically implemented for a small number of WAP Gateways belongs to a specific mobile operator (in Taiwan, most origin servers are associated with one WAP Gateway). Thus, if Algorithm II is implemented between the origin server and the WAP Gateway, the invalidation processing cost at the origin server is not a problem. On the other hand, to implement Algorithm II between the WAP Gateway and the WAP handsets, processing power and transmission power should be considered in the WAP Gateway for energy conservation at the WAP handset.

4.2. Invalid period in poll-each-read

Define the *invalid period* as the period between the instant that an update occurs and the instant that the next access arrives. In figure 4, the invalid period is $(t_3 - t_2)^+ = (\tau_2 - \tau_1)^+$. We notice that when there is no update between two accesses (which is the case when $t_3 < t_2$, the cached item is valid, thus the invalid period in this case is zero. In the invalid period, the cached entry in the client unnecessarily occupies cache storage in Algorithm I if the entry is not replaced. In Algorithm II, the cached entry is invalidated as soon as the entry is updated, and the storage can be re-used during the invalid period. Thus it is anticipated that the cache utilization for Algorithm II is better than that for Algorithm I. Define $E[T_{\text{inv}}]$ as the expected invalid period between two data accesses. $E[T_{\text{inv}}]$ is derived as follows:

$$E[T_{\text{inv}}] = \int_0^\infty \int_0^\infty (\tau_2 - \tau_1)^+ r_u(\tau_1) f_a(\tau_2)\, d\tau_1\, d\tau_2.$$

Although we can derive more general case, for the simplicity, we concentrate on two special cases. When the updates form a Poisson process, τ_0 and τ_1 will be exponentially distributed with the same distribution $\lambda \exp(-\lambda\tau_1)$, hence we have

$$\begin{aligned} E[T_{\text{inv}}] &= \int_{\tau_2=0}^\infty \int_{\tau_1=0}^{\tau_2} (\tau_2 - \tau_1)\lambda e^{-\lambda\tau_1} f_a(\tau_2)\, d\tau_1\, d\tau_2 \\ &= \frac{1}{\mu} - \frac{1}{\lambda}(1 - f_a^*(\lambda)) \end{aligned} \tag{9}$$

Define θ as the portion of invalidation time in an inter access arrival time period, then

$$\theta = \frac{E[T_{\text{inv}}]}{E[\tau_2]}$$

and from (9)

$$\theta = 1 - \left(\frac{\mu}{\lambda}\right)[1 - f_a^*(\lambda)]$$

which is the same as (8). When the accesses form a Poisson process, τ_2 is exponentially distributed with probability density function $\mu \exp(-\mu\tau_2)$. Thus,

$$\begin{aligned} E[T_{\text{inv}}] &= \int_{\tau_1=0}^\infty r_u(\tau_1) \int_{\tau_1}^\infty [(\tau_2 - \tau_1) f_a(\tau_2) d\tau_2]\, d\tau_1 \\ &= \int_{\tau_1=0}^\infty r_u(\tau_1) \int_{\tau_1}^\infty \left[(\tau_2 - \tau_1)\mu e^{-\mu\tau_2} d\tau_2\right] d\tau_1 \\ &= \int_{\tau_1=0}^\infty r_u(\tau_1) \int_{\tau_1}^\infty \left[\frac{1}{\mu}\exp(-\mu\tau_1)\right] d\tau_1 \\ &= \frac{1}{\mu} r_u^*(\mu) = \frac{\lambda}{\mu^2}(1 - f_u^*(\mu)) \end{aligned} \tag{10}$$

and

$$\theta = \frac{\lambda}{\mu}(1 - f_u^*(\mu)).$$

If we plot the θ curves based on (8), then these curves are the same as the β curves in figure 8. We observe that Algorithm I may keep stale entry for a long time (and thus the cache is not efficiently utilized) if the access pattern is irregular (i.e., v is large) and μ/λ is small.

5. Adaptive algorithm for single-level cache

From (1) and (2), we can derive when Algorithm I outperforms Algorithm II and vice versa.

$$\begin{aligned} C_I > C_{II} &\Leftrightarrow \alpha(y - x) + 2x > \alpha(3x + y) \\ &\Leftrightarrow 2x > 4\alpha x \\ &\Leftrightarrow \alpha < \frac{1}{2} \end{aligned} \tag{11}$$

Based on (11), we propose an adaptive algorithm to dynamically switch the cached access algorithm exercised between the server and the client. In this adaptive algorithm, the server keeps two counters for a data entry. Define a cycle as the period between two consecutive data accesses. In figure 4, the period $[t_1, t_3]$ is a cycle. Figure 9 shows four cycles, where updates occur in Cycles 1 and 3. The counter n_u measures the number

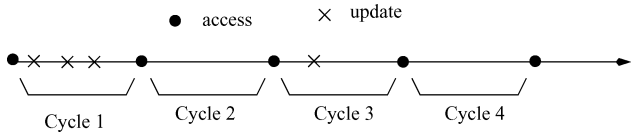


Figure 9. Updates in cycles.

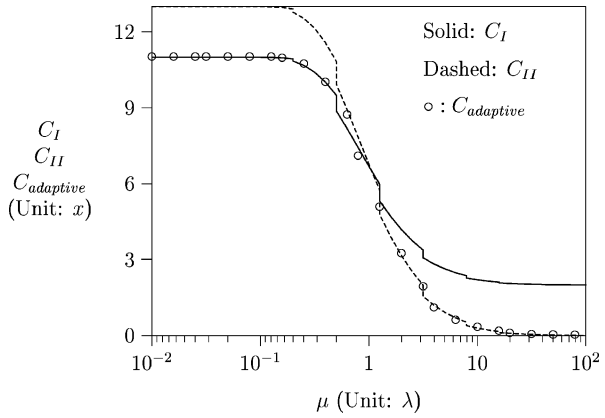


Figure 10. Comparison of the adaptive algorithm and Algorithms I and II.

of the cycles that have updates in the cycles, and the counter n_c measures the number of the cycles in an observed period. Thus, during an observed period,

$$\alpha = \frac{n_u}{n_c} \tag{12}$$

If the observed period is the four cycles in figure 9, then $n_u = 2$, $n_c = 4$, and $\alpha = 0.5$. When Algorithm I is exercised, the server increments n_c by one at Step I.3 (when the server receives an entry access message from the client), and increments n_u by one at Step I.3.2 (when the server sends the data entry to the client). When Algorithm II is exercised, every cached entry in the client keeps a count n_c^* to record the number of accesses since the last update. At Step II.3, if the cached entry exists, the client uses the cached entry and increments n_c^* by one. At Step II.1, the server increments n_u by one if an invalidation message is sent to the client. At Step II.2 (an update occurs), the client sends the n_c^* value to the server through the acknowledgment message, and then sets n_c^* to 0. When the server receives the acknowledgment from the client, it increments n_c by the amount n_c^*. When n_c is larger than a predefined value N_c, then the server computes α using (12), and determine which algorithm should be exercised based on (11). Then both counters n_c and n_u are reset. Our experiments indicate that when $N_c \geq 5$, the adaptive algorithm can accurately select the algorithm with lower communication cost. Figure 10 plots the communication costs for the adaptive algorithm (i.e., C_{adaptive}), Algorithms I and II. The n_c values in the adaptive algorithm is 10. The figure indicates good performance for the adaptive algorithm.

6. Analysis of multi-level cache

Suppose that there are $n > 1$ nodes in the data path between the origin server and the WAP handset (including these two nodes). Let node 0 be the origin server and node $n-1$ be the WAP handset. Suppose that node j ($1 \leq j \leq n-1$) has a cache and the cached data access algorithm $A[j]$ is exercised, where

$$A[j] = \begin{cases} I, & \text{for Algorithm I} \\ II, & \text{for Algorithm II} \end{cases}$$

There are two facts for multi-level cached data access algorithm:

Fact 2. Assume that $A[j] = A[j+1] = II$ ($0 < j < n-1$). When node j receives an invalidation message for a cached entry, node j should also sends an invalidation message to node $j+1$ if the status of the cached entry at node $j+1$ is valid.

It is clear that Fact 2 must hold, otherwise Fact 1 in Section 2 is violated.

Fact 3. For $0 < j < n-1$, if $A[j] = I$ then $A[j+1] = I$.

We prove by contradiction that Fact 3 holds. Assume that $A[j] = I$ and $A[j+1] = II$. When a data entry at node 0 is modified, the cached entry at node j is not modified until the next access of this entry occurs. Thus, no invalidation message for this entry is sent from node j to node $j+1$ and Fact 1 is violated at node $j+1$. Thus, the value for $A[j+1]$ must be I.

The communication cost $C[j]$ for an entry access between node j and node 0 is

$$C[j] = \begin{cases} 0, & j = 0 \\ C_{A[j]} + \alpha_j C[j-1], & 1 \leq j < n \end{cases} \tag{13}$$

where from (1) and (2)

$$C_{A[j]} = \begin{cases} \alpha_j(y_j - x_j) + 2x_j, & A[j] = I \\ \alpha_j(3x_j + y_j), & A[j] = II \end{cases} \tag{14}$$

In (14), x_j represents the communication cost for the entry access message or the validation/invalidation message between node j and $j-1$, y_j represents the communication cost for delivering a data entry from node $j-1$ to node j, and α_j represents the α value of the cache at node j.

Consider the two-level cache architecture in figure 1(b). Suppose that there are N WAP handsets connected to the WAP Gateway. The update rate is λ and the access rate at a WAP handset is μ. Let $C_{A[1],A[2]}$ be the $C[2]$ cost where the WAP Gateway exercises Algorithm $A[1]$ and the WAP handset exercises Algorithm $A[2]$. From (13), the cost $C_{A[1],A[2]}$ is expressed as

$$C_{A[1],A[2]} = \begin{cases} \alpha_2(y_2 - x_2) + 2x_2 + \alpha_2[\alpha_1(y_1 - x_1) + 2x_1], \\ \qquad A[1] = I,\ A[2] = I \\ \alpha_2(y_2 - x_2) + 2x_2 + \alpha_2\alpha_1(3x_1 + y_1), \\ \qquad A[1] = II,\ A[2] = I \\ \alpha_2(3x_2 + y_2) + \alpha_2\alpha_1(3x_1 + y_1), \\ \qquad A[1] = A[2] = II \end{cases} \tag{15}$$

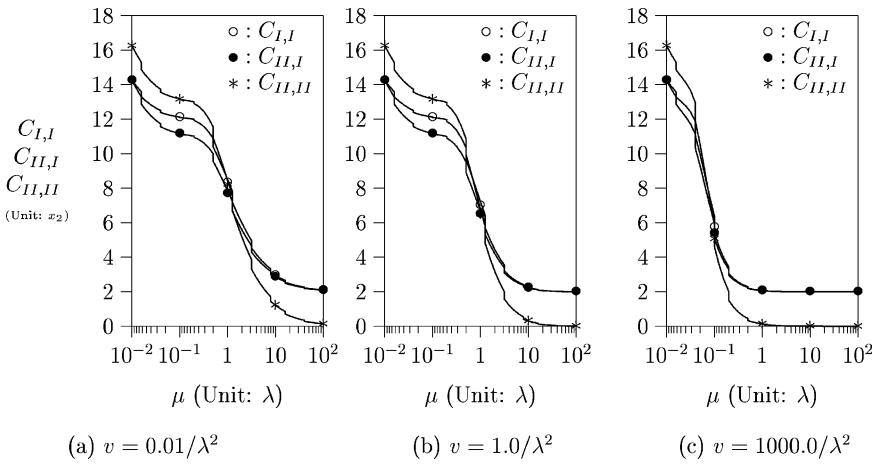


Figure 11. Performance of two-level cached data access algorithms ($N = 100$, $y_2 = 10x_2$, $x_1 = 0.5x_2$, $y_1 = 5x_2$).

The probabilities α_1 and α_2 are derived as follows. We only consider Case A. From (5), at the WAP handset, we have

$$\alpha_2 = 1 - f_a^*(\lambda)$$

At the WAP Gateway, if callback is exercised at the WAP handset, then the data access rate is

$$\mu_{II,\text{WAP-Gateway}} = \alpha_2 N \mu \tag{16}$$

because only α_2 of the accesses at a WAP handset will results in accesses at the WAP Gateway. If poll-each-read is exercised at the WAP handset, then the data access rate at the WAP Gateway is

$$\mu_{I,\text{WAP-Gateway}} = N\mu \tag{17}$$

If N is large enough, then the accesses to the WAP Gateway becomes a Poisson process [11]. Thus α_1 can be derived by using (16) and (5) for $A[2] = II$ and be serviced by using (17) and (5) for $A[2] = I$, where the inter access times are exponentially distributed ($v = 1$). Figure 11 shows the the communication costs for various two-level cached data access algorithms. In this figure, $N = 100$, $y_2 = 10x_2$, $x_1 = 0.5x_2$, and $y_1 = 5x_2$. This figure indicates that when the access rate μ is small, $C_{II,II} > C_{II,I}, C_{I,I}$. When the access rate is large, the result reverses. Furthermore, when the access pattern of a WAP handset is more irregular, the communication costs of the two-level cached data access algorithms become smaller. Figure 12 shows the impact of N on the performance of two-level cached data access algorithms. The figure indicates that the communication cost reduces as N increases. This result is consistent with what we observed before. As N increases, the access rate at the WAP Gateway increases, and it is more likely that WAP Gateway will have current copy of the data entry. For $N > 200$, the impact of N becomes less significant.

If $A[1] = II$, then the adaptive mechanism in Section 5 can be exercised between the WAP Gateway and the WAP handset to reduce the communication cost.

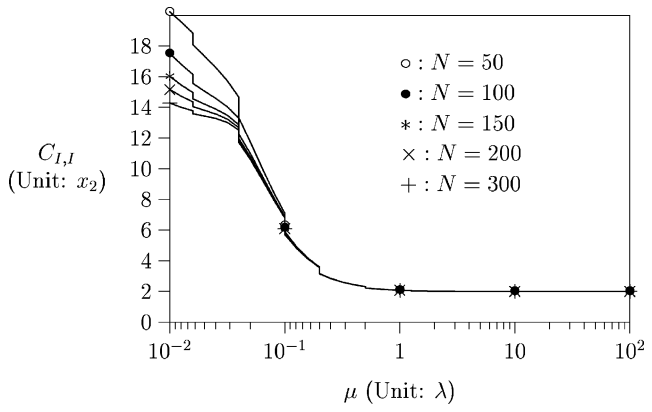


Figure 12. Performance of two-level cached data access algorithm (cont.; $A[1] = A[2] = I$, $y_2 = 10x_2$, $x_1 = 0.5x_2$, $y_1 = 5x_2$).

7. Conclusions

This paper investigated two cached data access algorithms for strongly consistent wireless data. In Algorithm I the client always asks the server if the cached entry in the client is valid when a data access occurs. In Algorithm II, the server always invalidates the cached entry in the client when an update occurs. We proposed an analytical model to derive the communication costs of these two algorithms. Let α be the probability that there are updates between two data access. Our derivation indicates that Algorithm II outperforms Algorithm I if and only if $\alpha < 1/2$. A small α occurs if the access rate is large compared with the update rate and/or the data access pattern is irregular (the variance of the inter access time distribution is large).

Based on the α statistics we designed an adaptive mechanism that switches between Algorithms I and II to reduce the communication cost for cached entry accesses. Our study indicates that this mechanism effectively selects the access algorithm with lower communication cost.

We also extended our algorithm for two-level cache hierarchy where the WAP Gateway has the first level cache and the WAP handsets have the second level caches. Our study provided guidelines to select data access algorithms under various traffic patterns. A WAP prototype (with push mechanism) has been developed in National Chiao Tung University. One of our future research directions is to implement the adaptive mechanism for the strongly consistent cached data access algorithms in our WAP prototype.

Acknowledgments

The authors would like to express their gratitude to the Editors and the reviewers for their valuable comments which have greatly enhanced the quality of this paper.

The work of Fang was supported in part by US National Science Foundation under Faculty CAREER Award ANI-0093241 and under grant ANI-0220287, and by the US Office of Naval Research under Young Investigator Award N000140210464 and under grant N000140210554. The work of Lin was sponsored in part by MOE Program for Promoting Academic Excellence of Universities under the grant number 89-E-FA04-1-4, FarEastone, IIS/Academia Sinica, CCL/ITRI, National Telecommunication development Program (NTP), and the Lee and MTI Center for Networking Research, NCTU.

References

[1] S. Asmussen, Matrix-analytic models and their analysis. Scandinavian Journal of Statistics 27(2) (2000) 193–226.

[2] I. Chlamtac, Y. Fang, and H. Zeng, Call blocking analysis for PCS networks under general cell residence time, in: *IEEE WCNC, New Orleans* (Sept. 1999).

[3] Y. Fang, and I. Chlamtac, Teletraffic analysis and mobility modeling for PCS networks. IEEE Transactions on Communications 47(7) (1999) 1062–1072.

[4] Y. Fang, I. Chlamtac, and H. Fei, Analytical results for optimal choice of location update interval for mobility database failure restoration in PCS networks. IEEE Transactions on Parallel and Distributed Systems (2000).

[5] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham and M. West, Scale and performance in a distributed file system, ACM Transactions on Computer Systems 6(1) (Feb. 1988) 51–58.

[6] N.L. Johnson, *Continuous Univariate Distributions-1* (John Wiley & Sons, 1970).

[7] N.L. Johnson, *Continuous Univariate Distributions-2* (John Wiley & Sons, 1970).

[8] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods in Stochastic Modeling* (SIAM, Philadelphia, 1999).

[9] Y.-B. Lin and I. Chlamtac, *Wireless and Mobile Network Architectures* (John Wiley & Sons, 2001).

[10] Y.-B. Lin, W.-R. Lai and J.-J. Chen, Effects of cache mechanism on wireless data access. IEEE Transactions on Wireless Communications (Accepted for Publication) (2003).

[11] I. Mitrani, *Modeling of Computer and Communication Systems* (Cambridge University Press, 1987).

[12] M. Nelson, B. Welch, and J. Ousterhout, Caching in the Sprite Network File System. ACM Transactions on Computer Systems 6(1) (Feb. 1988).

[13] S.M. Ross, *Stochastic Processes* (John Wiley & Sons) (1996).

[14] WAP Forum, Wireless application protocol architecture specification, Technical report, WAP Forum (1998).

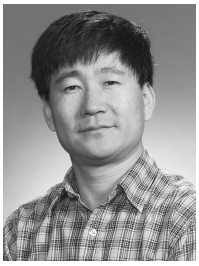
[15] WAP Forum, Wireless application protocol cache model specification, Technical report, WAP Forum (1998).

[16] WAP Forum, Wireless application protocol white paper, Technical report, WAP Forum (1999).

[17] WAP Forum, Wireless application protocol V1.1 to V1.2, Technical report, WAP Forum (1999).

[18] J. Yin, L. Alvisi, M. Dahlin and C. Lin, Volume leases for consistency in large-scale systems. IEEE Trans. on Knowledge and Data Engineering 11(4) (1999).

[19] J. Yin, L. Alvisi, M. Dahlin and A. Iyengar, Engineering server-driven consistency for large scale dynamic web services. ACM Transactions on Internet Technology 2(3) (2002) 224–259.



Yuguang Fang received the B.S. and M.S. degrees in Mathematics from Qufu Normal University, Qufu, Shandong, China, in 1984 and 1987, respectively, a Ph.D degree from Department of Systems, Control and Industrial Engineering at Case Western Reserve University, Cleveland, Ohio, in January 1994, and a Ph.D degree from Department of Electrical and Computer Engineering at Boston University, Massachusetts, in May 1997.

From 1987 to 1988, he held research and teaching positions in both Department of Mathematics and the Institute of Automation at Qufu Normal University. He held a post-doctoral position in Department of Electrical and Computer Engineering at Boston University from June 1994 to August 1995. From June 1997 to July 1998, he was a Visiting Assistant Professor in Department of Electrical Engineering at the University of Texas at Dallas. From July 1998 to May 2000, he was an Assistant Professor in the Department of Electrical and Computer Engineering at New Jersey Institute of Technology, Newark, New Jersey. From May 2000 to July 2003, he was an Assistant Professor in the Department of Electrical and Computer Engineering at University of Florida, Gainesville, Florida, where he has been an Associate Professor since August 2003. His research interests span many areas including wireless networks, mobile computing, mobile communications, automatic control, and neural networks. He has published over ninety papers in refereed professional journals and conferences. He received the National Science Foundation Faculty Early Career Development Award in 2001 and the Office of Naval Research Young Investigator Award in 2002. He is listed in Marquis *Who's Who in Science and Engineering*, *Who's Who in America* and *Who's Who in World*.

Dr. Fang has actively engaged in many professional activities. He is a senior member of the IEEE and a member of the ACM. He is an Editor for IEEE Transactions on Communications, an Editor for IEEE Transactions on Wireless Communications, an Editor for ACM Wireless Networks, an Area Editor for ACM Mobile Computing and Communications Review, an Associate Editor for Wiley International Journal on Wireless Communications and Mobile Computing, and an Editor for IEEE Wireless Communications. He was an Editor for IEEE Journal on Selected Areas in Communications: Wireless Communications Series and the feature editor for Scanning the Literature in IEEE Wireless Communications (formerly IEEE Personal Communications). He has also actively involved with many professional conferences such as ACM MobiCom'02, ACM MobiCom'01, IEEE INFOCOM'04, INFOCOM'03, INFOCOM'00, INFOCOM'98, IEEE WCNC'02, WCNC'00 (Technical Program Vice-Chair), WCNC'99, and International Conference on Computer Communications and Networking (IC3N'98) (Technical Program Vice-Chair).
E-mail: fang@ece.ufl.edu

Yi-Bing Lin received his BSEE degree from National Cheng Kung University in 1983, and his Ph.D. degree in Computer Science from the University of Washington in 1990. From 1990 to 1995, he was with the Applied Research Area at Bell Communications Research (Bellcore), Morristown, NJ. In 1995, he was appointed as a professor of Department of Computer Science and Information Engineering (CSIE), National Chiao Tung University (NCTU). In 1996, he was appointed as Deputy Director of Microelectronics and Information Systems Research Center, NCTU. During 1997-1999, he was elected as Chairman of CSIE, NCTU. His current research interests include design and analysis of personal communications services network, mobile computing, distributed simulation, and performance modeling. Dr. Lin has published over 150 journal articles and more than 200 conference papers.

Dr. Lin is a senior technical editor of IEEE Network, an editor of IEEE Trans. on Wireless Communications, an associate editor of IEEE Trans. on Vehicular Technology, an associate editor of IEEE Communications Survey and Tutorials, an editor of IEEE Personal Communications Magazine, an editor of Computer Networks, an area editor of ACM Mobile Computing and Communication Review, a columnist of ACM Simulation Digest, an editor of International Journal of Communications Systems, an editor of ACM/Baltzer Wireless Networks, an editor of Computer Simulation Modeling and Analysis, an editor of Journal of Information Science and Engineering, Program Chair for the 8th Workshop on Distributed and Parallel Simulation, General Chair for the 9th Workshop on Distributed and Parallel Simulation. Program Chair for the 2nd International Mobile Computing Conference, Guest Editor for the ACM/Baltzer MONET special issue on Personal Communications, a Guest Editor for IEEE Transactions on Computers special issue on Mobile Computing, a Guest Editor for IEEE Transactions on Computers special issue on Wireless Internet, and a Guest Editor for IEEE Communications Magazine special issue on Active, Programmable, and Mobile Code Networking. Lin is the author of the book Wireless and Mobile Network Architecture (co-author with Imrich Chlamtac; published by John Wiley & Sons). Lin received 1998, 2000 and 2002 Outstanding Research Awards from National Science Council, ROC, and 1998 Outstanding Youth Electrical Engineer Award from CIEE, ROC. He also received the NCTU Outstanding Teaching Award in 2002. Lin is an Adjunct Research Fellow of Academia Sinica, and is Chair Professor of Providence University. Lin serves as consultant of many telecommunications companies including FarEasTone and Chung Hwa Telecom. Lin is an IEEE Fellow.