IEICE TRANS. INF. & SYST., VOL.E94–D, NO.8 AUGUST 2011

PAPER

Probabilistic Broadcast-Based Cache Invalidation Scheme for Location Dependent Data in Mobile Environments

Shigeaki TAGASHIRA[†a)], ***Member***, **Yutaka KAMINISHI**[††], ***Nonmember***, **Yutaka ARAKAWA**[†], **Teruaki KITASUKA**[†††], ***and*** **Akira FUKUDA**[†], ***Members***

SUMMARY Data caching is widely known as an effective power-saving technique, in which mobile devices use local caches instead of original data placed on a server, in order to reduce the power consumption necessary for network accesses. In such data caching, a cache invalidation mechanism is important in preventing these devices from unintentionally accessing invalid data. In this paper, we propose a broadcast-based protocol for cache invalidation in a location-aware system. The proposed protocol is designed to reduce the access time required for obtaining necessary invalidation reports through broadcast media and to avoid client-side sleep fragmentation while retrieving the reports. In the proposed protocol, a Bloom filter is used as the data structure of an invalidation report, in order to probabilistically check the invalidation of caches. Furthermore, we propose three broadcast scheduling methods that are intended to achieve flexible broadcasting structured by the Bloom filter: fragmentation avoidance scheduling method (FASM), metrics balancing scheduling method (MBSM), and minimizing access time scheduling method (MASM). The broadcast schedule is arranged for consecutive accesses to geographically neighboring invalidation reports. In addition, the effectiveness of the proposed methods is evaluated by simulation. The results indicate that the MBSM and MASM achieve a high rate of performance scheduling. Compared to the FASM, the MBSM reduces the access time by 34%, while the fragmentations on the resultant schedule increase by 40%, and the MASM reduces the access time by 40%, along with an 85% increase in the number of fragmentations.

key words: *data caching, broadcast-based cache-invalidation, broadcast scheduling, probabilistic approach*

1. Introduction

The explosive spread of wireless network technology has rapidly increased the number of users who enjoy various promising ubiquitous services from anywhere and at any time. In particular, current mobile devices, such as cellular phones and smart phones with wireless communication devices, enable us to access an assortment of beneficial information in an outdoor environment, such as travel assistance, shopping guides, and so on. Our target in this paper is such geographical information services. In the current mobile technology, geographical information services are usually provided by the client-server approach. The technology has many advantages, such as making mobile devices smaller and lighter and always providing the latest information for mobile users. However, it also has the disadvantage of requiring a continuous Internet connection, which is the main factor in battery drainage. Therefore, power-saving technology is one of the most important issues for improving the convenience of geographical information services.

Data caching is widely known as an effective power-saving technique that stores frequently accessed data into local storage on a mobile device in advance. It has been shown that data caching can reduce network traffic by accessing already stored data preferentially [1]. Furthermore, it has been applied to various network architectures such as ad hoc networks [2], Internet-based mobile ad hoc networks [3], [4], and wireless cellular networks [5]. Although such data caching can decrease the power consumption necessary for network access, the mobile device may unfortunately access invalid (past) information. Therefore, we need to invalidate such caches, in order to prevent the mobile device from unintentionally accessing invalid information.

Existing cache-invalidation methods for mobile environments are focused on several design aspects [6]. From the aspect of consistency control, these methods can be mainly classified into two types: push-type and pull-type. In a push-type (broadcast-based) protocol, a server can deliver invalidation reports to mobile devices (clients) with a low communication cost, regardless of the number of clients. In addition, the power consumption required for checking the invalidation reports is low, because the client can check only by receiving the reports from the server through broadcast media. In general, the power consumption required for receiving data is lower than that for sending data. A client passively listens to a report for cache invalidation that is broadcasted by the server. However, at the time when the client accesses its own cache, the invalidation report associated with the cache is not always broadcasted. The client must wait for the necessary report on the broadcast schedule to maintain consistency. Conversely, a pull-type protocol needs to access the server for a consistency check in a periodical manner or in an on-demand manner. Because the load of the server will increase in proportion to the number of clients, the scalability of this protocol is low. Geographical information services are generally used by many users, and the number of users is changeable; therefore, a broadcast-based invalidation protocol is suitable for such services.

In this paper, we consider data caching in geographical information services and propose broadcast-based cache-

Manuscript received January 6, 2011.
Manuscript revised April 15, 2011.

[†]The authors are with the Graduate School/Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka-shi, 819–0395 Japan.
[††]The author is with the FUJITSU Kyushu Network Technologies, Fukuoka-shi, 814–8588 Japan.
[†††]The author is with the Graduate School of Science and Technology, Kumamoto University, Kumamoto-shi, 860–8555 Japan.
a) E-mail: shigeaki@f.ait.kyushu-u.ac.jp

DOI: 10.1587/transinf.E94.D.1590

invalidation protocols for such caching. The proposed protocols use access time and fragmentation as performance metrics: access time is the time elapsed between the moment when a client accesses its own cache and the moment when it receives the invalidation report associated with the cache, and fragmentation is the number of repeats between the active and sleep states in the communication device on the client when receiving the necessary reports. In the proposed protocol, a Bloom filter is adopted to expand and compress the invalidation report to be broadcasted, to shorten the access time. Furthermore, we propose three extended broadcast scheduling methods to achieve flexible broadcasting structured by the Bloom filter, in terms of access time and fragmentation: fragmentation avoidance scheduling method (FASM), metrics balancing scheduling method (MBSM), and minimizing access time scheduling method (MASM). The FASM and MASM are particularly focused on avoiding fragmentation and the access time, respectively. The MBSM aims to reduce the access time with fewer fragmentations. We evaluate the average access time and the number of fragmentations by computer simulation. The results indicate that, compared to the FASM, the MBSM can reduce the access time by 34% and increase the number of fragmentations by 40%. In addition, the MASM can reduce the access time by 40% and increase the number of fragmentations by 85%.

The rest of this paper is organized as follows: Sect. 2 introduces the existing cache-invalidation methods based on the push mechanism. Section 3 describes our system model and issues for cache invalidation in the system model. Section 4 describes the proposed mechanism and Sect. 5 describes our evaluation of the mechanism. Finally, Sect. 6 lists some conclusions and directions for future research.

2. Related Study

In this section, we introduce existing push-type invalidation methods. As a stateless cache-management approach, the broadcasting timestamps (TS) method, which uses a timestamp, has been proposed in [1]. In this method, a mobile support station (MSS) is placed as a stationary host designed to manage caches on clients. An MSS broadcasts an invalidation report (IR) to the connected clients every L seconds. An important issue in this method is an L-second delay caused when receiving the IR in a worst-case scenario. Furthermore, because an IR records only the timestamp of the original data that was updated in the past $w(> L)$ seconds, the client needs to invalidate the entire cache if it cannot connect to the MSS in over w seconds.

An updated invalidation report (UIR) has been proposed in [7] to address the delay issue in the TS method, which is also a push-based stateless management approach. A UIR records the timestamp of each item of the original data updated after the last IR. By broadcasting the UIR more than once between the IR cycles, it reduces the delay in receiving the IR. However, if the frequency of updates increases, the size of the broadcast messages becomes large, which wastes network capacity. In addition, the increase in communication frequency wastes the battery. As an alternative extension to the IR, a dynamic invalidation report (DIR) has been proposed in [8]. In a DIR, an early validation mechanism is employed to reduce the long latency in IR-based schemes.

As another approach, an asynchronous stateful (AS) method has been proposed in [9]. This protocol is a stateful push-type protocol, that is, the MSS comprehends caches stored on connected clients. In the AS method, because the MSS manages the state for caches stored on each connected client, it can send to only the specific clients for which the caches of the original data have been updated. Therefore, the AS method achieves a reduction in meaningless broadcasts as well as complete consistency in an error-free environment. In addition, even if a client cannot connect to the MSS for over w seconds, it does not need to invalidate any cache. However, an increase in the number of clients severely concentrates the load of the MSS, which leads to a lack of scalability. It is also difficult to completely comprehend the state of the clients' caches because of topological changes frequently experienced in such wireless communication environments. In another approach [10], a cache invalidation algorithm, called bit-sequences (BS), has been proposed. The periodical IR consists of a set of binary bit sequences with an associated set of timestamps. In the BS, even if a disconnection period lasts over w seconds, it does not need to invalidate all of the caches.

To deal with a long disconnection of clients, a selective adaptive sorted (SAS) cache invalidation strategy and an enhanced scalable asynchronous cache consistency scheme (ESACCS) has been proposed in [11] and [12], respectively. In the SAS method, an invalidation report includes a dynamic history of updates, i.e., an initial report covers the minimum history of updates to avoid long latency, but this history can be enlarged according to the clients' request. The ESACCS takes a hybrid approach between the stateless and stateful approaches. In the ESACCS method, each cache in the clients has a TTL. If a cache is expired, the cache is marked as an uncertain state, which does not ensure its consistency (does not determine whether valid or invalid). By using an uncertain state, when a client cannot connect to the MSS in over w seconds, it avoids deleting all caches, i.e., it checks the consistency of the MSS in an on-demand manner.

As mentioned above, these approaches are not intended for geographical information services. In this paper, we consider a push-type invalidation method especially for location dependent data.

For location-dependent data, in [13], the bit vector with compression (BVC), grouped bit vector with compression (GBVC), and implicit scope information (ISI) methods have been proposed. The proposed methods address the cache-invalidation issue caused by both temporal dependency and location dependency. In particular, cache inconsistency can be caused by locations changing in location-dependent invalidation; the valid scope of a cache is defined as a set of

areas where the cache is valid and the cache must be invalidated out of its scope. In [14], the valid scope of a location-dependent cache is defined as positions where queries issued from clients would likely return the same results. It can support location-dependent spatial queries such as NN queries, kNN queries, range queries and window queries. Although these approaches focus on the valid scope of a location-dependent cache, they do not consider the distribution of invalidation reports. In this paper, we propose an efficient scheduling scheme in broadcast-based invalidation protocols for location dependent data.

3. Cache Invalidation for Geographical Data

3.1 System Model

Figure 1 shows our system model. In this paper, our target geographical information services are traffic information service, shopping information service, and so on. In such services, a client retrieves geographical data on its surroundings via a communication media or a broadcast media from a server. The client caches the retrieved data in its local storage, and can reuse the caches if the original data are not updated, which leads to the reduction of client's power consumption for network accesses. In this model, when a client accesses its own cache, it has to check the invalidation report for the cache received through a broadcast media, i.e., whether the original data of the cache is updated or not. A broadcast media can realize the distribution of invalidation reports with a low communication cost, regardless of the number of clients. In addition, a client can check invalidations by passively listening to reports that are broadcasted by the server. The client's power consumption required for receiving invalidation reports is low, because the power consumption in receiving mode is generally lower than that in transmission mode. If an update exists, the client gets its latest data from the server and uses it. Otherwise, the client uses the cache. Note that in this paper, we will focus on the distribution of not geographical data but invalidation reports.

A service region for our system is partitioned into several small areas. An invalidation report is associated with an area, i.e., the invalidation report for an area includes a set of invalidations for geographical data mapped to the area. More specifically, geographical data, based on latitude and longitude, is mapped onto a $2^n \times 2^n$ meshed area, where n is an integer. Let $I(x, y, 0)$ denote an invalidation report for a set of geographical data mapped on a basic area $(x, y, 0)$ $(x, y = 0, 1, \ldots, 2^n - 1)$. More generally, let $I(x, y, p)$ $(p \leq n)$ be an invalidation report for a $2^p \times 2^p$ square area (x, y, p), i.e., $I(x, y, p) = \bigcup_{i=0}^{2^p-1} \bigcup_{j=0}^{2^p-1} I(x+i, y+j, 0)$. For example, we can define $I(0, 0, n)$ as an invalidation report for geographical data in the entire area of this system. The invalidation reports are broadcasted in an order of their associated areas, which is called the **broadcast schedule**. The scheduling method decides the broadcast schedule in terms of the following two performance metrics.

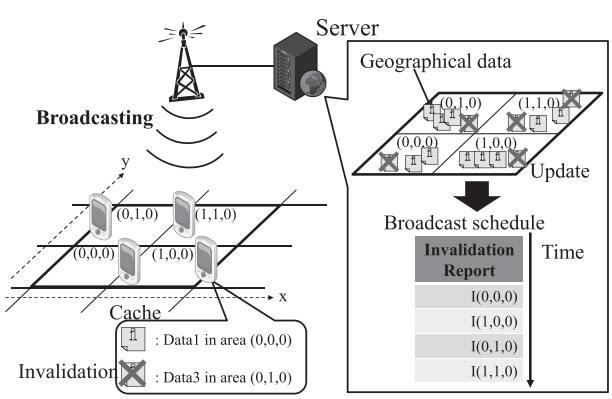


Fig. 1 Cache invalidation with broadcast-based protocol.

3.2 Performance Metrics

In this section, we describe two performance metrics for our proposed broadcast-based invalidation mechanism.

- In a broadcast-based protocol, a client needs to receive the broadcasted invalidation report when it uses its own cache. However, the invalidation report associated with the cache is not always broadcasted. The client must wait for the report on the broadcast schedule. In this paper, the term "**Access time**" is defined as the period from the time when a client uses its own cache to the time when it receives the invalidation report associated with the cache. In general, the average access time is about $T/2$, where the broadcast cycle is T. Figure 2 shows an example. In this example, a service region is divided into 4×4 areas. The client tries to use the cache of data1 which is mapped on area $(3, 3, 0)$. It has to wait for receiving $I(3, 3, 0)$ to check the invalidation of the cache, although the server is broadcasting $I(0, 0, 0)$.
- If the broadcast schedule is delivered to a client in advance, it is not necessary for the client to always monitor broadcasted invalidation reports. Because the client can be in sleeping mode during the period for broadcasting its unnecessary reports, it can decrease the power consumption required for listening and receiving the data. However, if some necessary invalidation reports are dispersed on the broadcast schedule, the client needs to repeat the start-up and the sleep-down within one broadcasting cycle. We define the condition in which parts of the necessary invalidation reports are dispersively assigned on the broadcast sched-

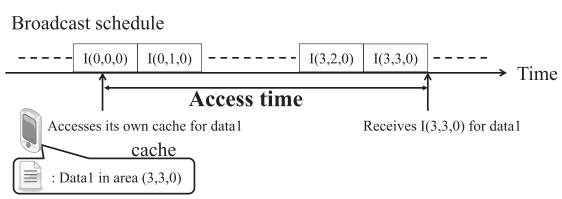


Fig. 2 Example of maximum access time.

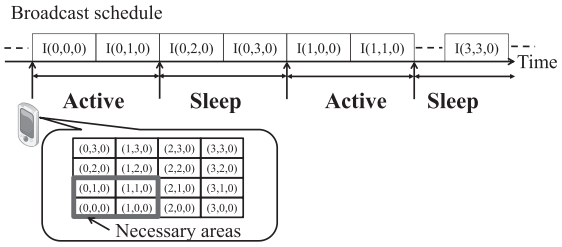


Fig. 3 Example of fragmentation.

ule as "**fragmentation**". Figure 3 shows that fragmentation occurs because the invalidation reports on the geographically close areas (that is, $I(0,0,0)$, $I(0,1,0)$, $I(1,0,0)$, and $I(1,1,0)$) are dispersively assigned on the schedule.

In a broadcast-based cache-invalidation protocol, it is important to reduce both the access time and fragmentation. Therefore, we propose a novel cache-invalidation protocol that takes both functions into account.

The reduction of the access time can be realized by asking the updates of cached data to the server through the communication media in either a periodical manner or an on-demand manner, and broadcasting invalidation reports only for updated and cached data, that is, only required invalidation reports are broadcasted in a variable-length schedule. However, an increase in the number of clients severely concentrates the load of the server, which leads to a lack of scalability. It also degrades the energy saving performance due to the transmissions for asking the updates. In this paper, we focus on a stateless broadcast-based cache-invalidation protocol which periodically broadcasts invalidation reports for all geographical data.

4. Proposed Broadcast-Based Cache-Invalidation Protocol

Our proposed protocol adopts an efficient data structure for expressing cache-invalidation reports. In addition, we propose three broadcast scheduling methods to support the broadcasting which is structured by a Bloom filter. The key points of our ideas are as follows:

- To reduce the access time, we adopt a Bloom filter as a data structure for an invalidation report. Although it probabilistically checks the invalidation of caches, the size of an invalidation report can be compressed significantly and thus, a cycle for broadcasting is decreased by reducing the total size of the invalidation reports, which leads to a reduction in the average access time.
- We propose three scheduling methods for each purpose: the FASM can avoid fragmentation, the MBSM can improve the access time with fewer fragmentations, and the MASM can reduce the access time.

4.1 Data Structure of Cache-Invalidation Report

The Bloom filter, which was proposed by Burton H. Bloom in 1970, is a stochastic structure with high space efficiency [15]. It has been widely applied to many application fields [16], [17]. Let U be a universe of elements and S be a subset of U. The objective of the Bloom filter is to check whether each element in U is contained in S by using a bit sequence B of length m. It uses k different hash functions $h_1, h_2, \ldots, h_k$ from U to set $\{1, 2, \ldots, m\}$. For each element i in set S, it calculates k hash values $h_1(i), h_2(i), \ldots, h_k(i)$, and it then sets the $h_j(i)$-th bit of B to 1 for each $1 \leq j \leq k$. While checking, it checks whether a given element $u \in U$ is contained in S. It calculates k hash values $h_1(u), h_2(u), \ldots, h_k(u)$. If the $h_j(u)$-th of B is equal to 1 for all $1 \leq j \leq k$, u is included in set S. However, due to a conflict of hash values, u may not be included in set S even if all the bits are 1. Such false detection is generally called false positive, and its incidence rate fp is represented as follows:

$$fp = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k, \tag{1}$$

where the number of elements is n. On the other hand, although a Bloom filter may return a false positive, it does not allow a false negative, i.e., u is never included in set S if at least one of the bits indicated by the hash values is not 1.

Following is a description of the use of the Bloom filter in our proposed protocol. Let V denote a set of invalidations for an area, and let $bf(V)$ denote a Bloom filter for V. A client can obtain $bf(V)$ as an invalidation report as a broadcast from the server. By using $bf(V)$, the client can check whether its own cache u for the area is updated, i.e., u is included in set V. If no update exists, the client can immediately use its cache, because the Bloom filter never reports a false negative. If it considers that an update may exist, it directly requests the server to check the update. When the existence of the update is confirmed, the client obtains the updated information from the server and uses it.

4.2 Broadcast-Scheduling Method for Cache-Invalidation Reports

In this section, we explain the three proposed scheduling methods: FASM, MBSM, and MASM.

4.2.1 FASM

The main purpose of the FASM is to avoid fragmentation. We focus on the client's access pattern to geographical information and reflect the pattern to the broadcast scheduling. In many geographical information services, a client usually needs reports that include its geographically neighboring areas. Therefore, we schedule such areas' invalidation reports to be broadcasted in a continuous manner.

```
procedure FASM
B ← sche(F_0(x, y, n));
end procedure

{This function calculates schedule of (r, s, q) in z × z size. }
function sche(F_z(r, s, q))
Initialize L;
if q > z then
   p ← 2^{q−1};
   Enqueue L ← sche(F_z(r, s, q − 1));
   Enqueue L ← sche(F_z(r, s + p, q − 1));
   Enqueue L ← sche(F_z(r + p, s, q − 1));
   Enqueue L ← sche(F_z(r + p, s + p, q − 1));
   return L;
else
   return I(r, s, q);
end if
end function
```

Fig. 4 Pseudo code for FASM.

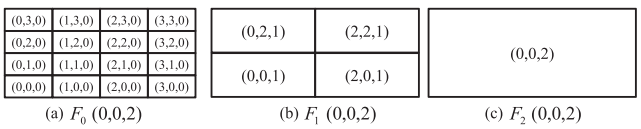


Fig. 5 Examples of filtering size for 4 × 4 area.

More specifically, we show a concrete algorithm for the FASM. Figure 4 shows the pseudo code for the FASM. In this code, the input is a whole area (x, y, n), and the output is queue B, i.e., the elements included in B are broadcasted in their order, in the form of a Bloom filter. The schedule is recursively decided by the function *sche*(), as shown in Fig. 4.

The function *sche*() calculates the schedule of an input $F_z(r, s, q)$, in which the schedule of a given area (r, s, q) is composed of $2^z \times 2^z$ areas. In this paper, the size of this area $2^z \times 2^z$ is called the **filtering size**. In other words, because a Bloom filter is assigned to each area with the filtering size in this method, the filtering size represents the size of an area which is filtered by one Bloom filter. Figure 5 shows examples of filtering size for the entire area 4×4. In Fig. 5 (a), $F_0(0, 0, 2)$ represents 16 areas for realizing 1×1 filtering size and $sche(F_0(0, 0, 2))$ calculates the schedule of 16 areas. Similarly, in Fig. 5 (b), $F_1(0, 0, 2)$ shows four areas with 2×2 filtering size, and $sche(F_1(0, 0, 2))$ calculates the schedule of four areas, e.g., $(0, 0, 1)$, $(0, 2, 1)$, $(2, 0, 1)$, and $(2, 2, 1)$. It is worth noting that, as the filtering size increases, the total number of areas for covering the entire area decreases. Therefore, the total length of the schedule decreases. However, this leads to an increase in the ratio of false positives, because the number of elements (or the number of items of an invalidation report) included in one Bloom filter increases, as described in Sect. 4.1. Therefore, there is the trade-off between the filtering size and the ratio of false positives. In FASM, the filtering size is fixed to 1×1 (that is, the basic area).

We show the behavior of the algorithm by using a concrete example. Let $n = 2$, that is, suppose the broadcast schedule for area $(0, 0, 2)$. Figure 6 shows the scheduling process of the FASM. The schedule B of area $(0, 0, 2)$ is recursively calculated as $sche(F_0(0, 0, 2)) \rightarrow [sche(F_0(0, 0, 1)),\ sche(F_0(0, 2, 1)),\ sche(F_0(2, 0, 1)),\ sche(F_0(2, 0, 1))] \rightarrow [I(0, 0, 0), I(0, 1, 0), I(1, 0, 0), I(1, 1, 0), I(0, 2, 0),\ I(0, 3, 0),\ I(1, 2, 0),\ I(1, 3, 0),\ I(2, 0, 0),\ I(2, 1, 0), I(3, 0, 0), I(3, 1, 0), I(2, 2, 0), I(2, 3, 0), I(3, 2, 0), I(3, 3, 0)]$.

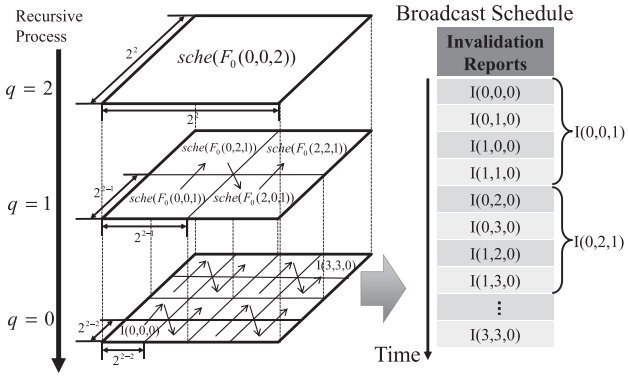


Fig. 6 Recursive procedure of FASM for 4 × 4 area.

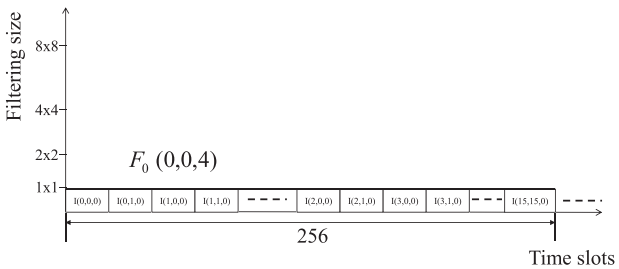


Fig. 7 Example of broadcast schedule for 16 × 16 area in FASM.

With this scheduling, the client can sequentially (without fragmentation) receive an invalidation report for any area included in set $C(x, y, n)$ for $I(x, y, n)$: $C(x, y, n) = \bigcup_{k=0}^{n} \bigcup_{l_x=0}^{2^{[n-k]}-1} \bigcup_{l_y=0}^{2^{[n-k]}-1} I(l_x \times 2^k + x, l_y \times 2^k + y, k)$, that is, in the case of $C(0, 0, 2)$, this set includes $I(0, 0, 2)$, $I(0, 0, 1)$, $I(0, 2, 1)$, $I(2, 0, 1)$, and $I(2, 2, 1)$, in addition to invalidation reports for all the 1×1 basic areas. For example, when the client needs the invalidation report for $I(0, 0, 1)$, it consecutively receives from $I(0, 0, 0)$ to $I(1, 1, 0)$ on the broadcast schedule.

Here, we illustrate an overall representation of the schedule for the FASM, as shown in Fig. 7. This figure shows a representation of the broadcast schedule for a 16×16 area in the FASM. In this paper, we define a **time slot** as a time block required for broadcasting one Bloom filter. In the FASM, all elements are broadcasted in a 1×1 filtering size, according to the resultant schedule derived from $sche(F_0(0, 0, 4))$. Thus, the total time slots required for one broadcast cycle is 256 slots.

4.2.2 MBSM

The MBSM is an extended protocol of the FASM, which can

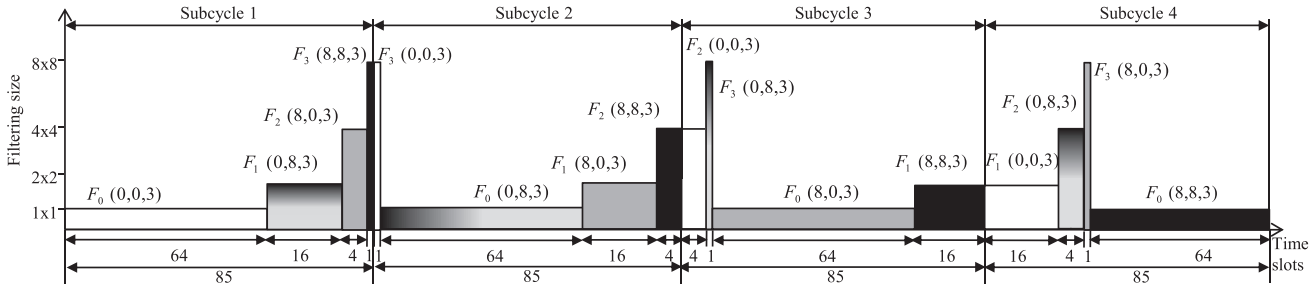


Fig. 8 Example of broadcast schedule for 16 × 16 area in MBSM (i = 2).

reduce the access time and retain the consecutiveness of the broadcast schedule of the surrounding areas. The MBSM divides the broadcast cycle into $2^i (i < n)$ subcycles and the whole area into 4^j groups, where j is an integer satisfying $4^{j-1} < 2^i \leq 4^j < 2^n$. If j that satisfies the condition does not exist, the schedule cannot be determined by the MBSM. In the MBSM, the filtering size to be broadcasted is different in each group, that is, although the filtering size is fixed to a basic area in the FASM, it is varied along with each subcycle for each group in the MBSM.

Figure 8 illustrates an overview representation of the broadcast schedule for a 16 × 16 area when $i = 2$. In this example, the broadcast cycle is divided into four subcycles, and the entire area $(0, 0, 4)$ is partitioned into four groups: $(0, 0, 3)$, $(0, 8, 3)$, $(8, 0, 3)$, and $(8, 8, 3)$. We find that the invalidation reports are broadcasted in different slots and with different filtering sizes; for example, the filtering size for group $(0, 0, 3)$ is 1×1, 8×8, 4×4, and 2×2 in subcycle 1, 2, 3, and 4, respectively. In addition, in group $(0, 0, 3)$ on subcycle 1, the filtering size is 1×1 (that is, $F_0(0, 0, 3)$) and the total number of time slots is 64 for the group. As for group $(0, 8, 3)$ on subcycle 1, the area size of the group is the same as that of group $(0, 0, 3)$, i.e., 8×8 area, the filtering size is 2×2 (that is, $F_1(0, 8, 3)$), and thus, the total number of slots is 16. Therefore, we can decrease the number of time slots as compared with the first group, although the area size is the same between these groups.

The MBSM can drastically reduce the access time if no update exists. In addition, the effectiveness of the reduction can be improved by using the Bloom filter of a filtering size larger than 1×1; that is, if the number of subcycles 2^i is increased (that is, by using the Bloom filter for a larger filtering size), we can reduce the total number of broadcasted invalidation reports for the whole area and thus reduce broadcast cycles. As a result, the access time of the clients is also reduced. However, this leads to an increase in the ratio of false positives, as described in Sect. 4.2.1. Therefore, the effectiveness of the MBSM strongly depends on the balance between the two conflicting factors.

Next, we show a concrete algorithm for MBSM. Figure 9 shows the pseudo code for the MBSM. The whole area (x, y, n) and the number of subcycles 2^i are provided as inputs to this procedure. The resultant schedule of this procedure is stored in queue B. First, the input area is divided into 4^j groups by the function *mkgroup*(). The divided groups

```
procedure MBSM
{The number of subcycles is 2^i.}
{The number of groups is 4^j.}
for θ = 0 to 2^i − 1 do
   G ← mkgroup(F_0(x, y, n));
   {Calculates schedule for each group in G.}
   while G is not empty. do
      Dequeue F_z(r, s, q) ← G;
      z ← z − θ + 2^i   (mod 2^i);
      B ← sche(F_z(r, s, q));
   end while
end for
end procedure

{Makes 4^j groups and returns scheduling data for each group.}
function mkgroup(F_z(r, s, q))
Initialize L;
if 2^i > 4^{n−q} then
   p ← 2^{q−1}; z ← z × 4;
   Enqueue L ← mkgroup(F_z(r, s, q − 1));
   Enqueue L ← mkgroup(F_{z+1}(r, s + p, q − 1));
   Enqueue L ← mkgroup(F_{z+2}(r + p, s, q − 1));
   Enqueue L ← mkgroup(F_{z+3}(r + p, s + p, q − 1));
   return L;
else
   return F_z(r, s, q);
end if
end function
```

Fig. 9 Pseudo code for MBSM.

are stored into queue G. Each element in G is scheduled by the function *sche*(), according to the specified filtering size, as in the FASM. In addition, the filtering size for a group can be changed by increasing the phase parameter θ.

We explain the behavior of the clients. When the client receives the invalidation report of the 1×1 filtering size, the process is the same as in the FASM. However, it is different when the client receives the report for a larger filtering size. In such an area, it is considered that the ratio of false positives is relatively high, compared with that of the 1×1 filtering size. Therefore, if it detects no update, it immediately uses the cache. Otherwise, it waits for the next report on broadcasting, using a smaller filtering size whose false-positive ratio is lower, without directly requesting the server to check the update. The client then continues this process until receiving the report of the 1×1 filtering size.

4.2.3 MASM

The MASM aims to decrease the average access time. In general, the average access time depends on the broadcast cycle for a client's necessary area. Therefore, we propose a preemptive scheduling method where the broadcasting period for each group is uniformly assigned on the timeline, while the filtering size is varied.

Figure 10 illustrates an overall representation for the broadcast schedule for a 16×16 area when $i = 2$. This schedule is similar to that of the MBSM; Although the total period for each group in one subcycle is the same as that of the MBSM, the broadcasting period of a group can be intercepted by that of another group. Therefore, we find that the continuous period assigned for each group is short, and thus the fragmentation increases, as compared to the MBSM.

In particular, we show a concrete algorithm for the MASM. Figure 11 shows the pseudo code for MASM. The entire area (x, y, n) and the number of subcycles 2^i are provided as inputs to this procedure. The resultant schedule of this procedure is stored in queue B. This code for the MASM is similar to that for the MBSM. The unique aspect for the MASM is to store the schedule for each group into an individual queue. After that, the final schedule is obtained by merging the elements in these queues in proportion to the size of each queue.

When the client receives the invalidation report, the process is same as in the MBSM. The MASM achieves a reduction in the average access time by uniformly assigning invalidation report for each group. Because this assignment is achieved using Bloom filters for not only the 1×1 filtering size but also a larger one, it is important to decide the optimal filtering size for each group to achieve good performance with the MASM. Furthermore, it is remarkable that the FASM's sequentiality is lost by the MASM.

4.3 Extensions of Proposed Schemes

In this section, we describe two extensions to the proposed schemes: grid adjustment and the variable length of the Bloom filter. Following are explanations of the extensions to the proposed schemes.

4.3.1 Grid Adjustment

In the proposed schemes, an input map is divided into $2^n \times 2^n$ areas and each item of invalidation information for geographical information is mapped onto one of the divided areas according to its latitude and longitude, as described in Sect. 3.1. In addition, an invalidation report for each area is expressed in the form of the fixed length of a bit sequence by using a Bloom filter. However, in the real world, the geographical distribution of information is nonuniform. Therefore, the ratio of false positives is different in each area.

To avoid the unbalanced ratio of false positives among areas, we propose an extension to the proposed scheme that adjusts the size of areas to balance the amount of geographical information among the areas, while the number of areas remains as identical as possible. In particular, we expand or shrink the width and/or height of the coordinate grid used for the division of an input area, in such a way that the divided areas are almost equal in terms of the amount of geographical information.

```
1: procedure MASM
2: Initialize B;
3: for θ = 0 to 2^i − 1 do
4:    G ← mkgroup(F_0(x, y, n));
5:    {Stores schedule for each group.}
6:    k ← |G|;
7:    for m = 1 to k do
8:       Dequeue F_z(r, s, q) ← G;
9:       z ← z − θ + 2^i  (mod 2^i);
10:      C_m ← sche(F_z(r, s, q));
11:   end for
12:   {Merges C_1,C_2,...,C_k into B in proportion to queue size.}
13:   p ← max(|C_1|, |C_2|, ..., |C_k|);
14:   for l = 1 to p do
15:      for m = 1 to k do
16:         if l  (mod p/|C_m|) is equal to 0. then
17:            Dequeue e ← C_m;
18:            Enqueue B ← e;
19:         end if
20:      end for
21:   end for
22: end for
23: end procedure
```

Fig. 11 Pseudo code for MASM.

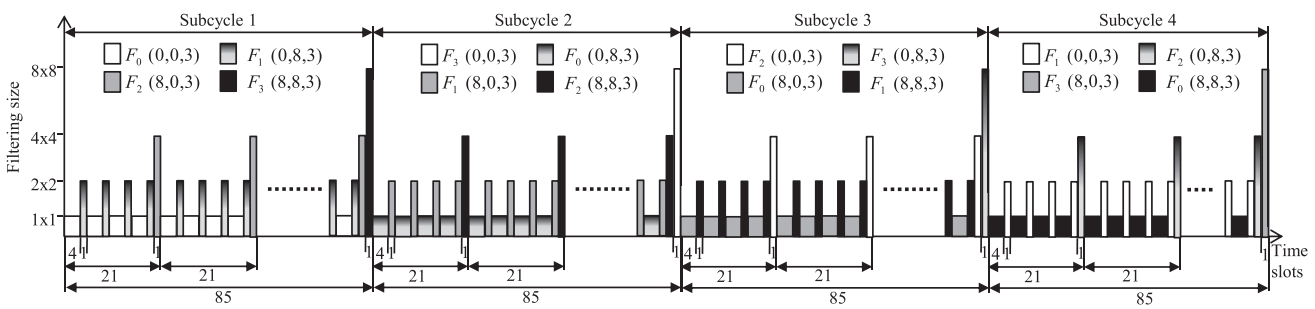


Fig. 10 Example of broadcast schedule for 16×16 area in MASM ($i = 2$).

4.3.2 Variable Length of Bloom Filter

Both the MBSM and the MASM use a bit sequence of the Bloom filter to express invalidation reports by using the Bloom filter, for not only the 1×1 filtering size but also for a larger one. As the filter size expands in a Bloom filter, the ratio of false positives increases. Therefore, we propose an extension of the variable length of the bit sequence in the Bloom filter, which assigns a longer bit sequence against a Bloom filter that includes a larger filtering size. More specifically, we use a $(p + 1) \times m$ bit sequence of the Bloom filter for a $2^p \times 2^p$ filtering size. Although the broadcast cycle slightly increases by increasing the length of the bit sequence, the ratio of false positives decreases, which leads to a reduction in the access time.

5. Evaluation

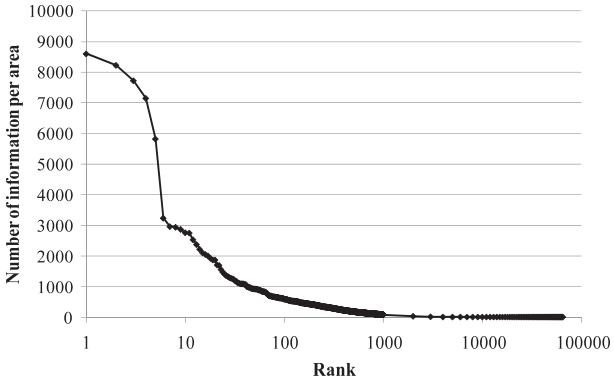
In this section, we evaluate our proposed methods FASM, MBSM, and MASM by computer simulations, in terms of average access time and fragmentation.

5.1 Simulation Environment

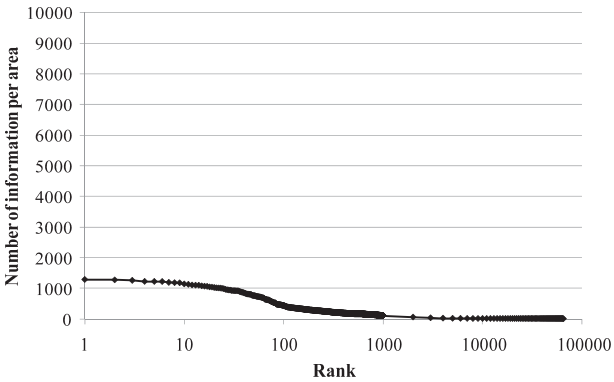
In this simulation, we use the information of facilities as the geographical data. This information is mapped on a 2-D mesh that represents the latitude and longitude. Each mesh is divided into a 32-second (approximately 1 km) scale based on its coordinates. A mesh divided into a 32-second scale has been defined in [18] as the search mesh, which can cover the entire world using four-byte expressions. More concretely, we use the information of 1,052,890 facilities that exist in an actual city in Japan. The facilities' information is mapped onto 256×256 areas, according to its latitude and longitude. Figure 12 (a) shows the geographical distribution of the facilities' information. In this figure, the horizontal axis represents the areas ranked according to the number of facilities' information, and the vertical axis represents the number of facilities' information per area. From this figure, we can observe that the distribution of facilities' information is nonuniform, and the number of facilities' information per area is inversely proportional to its rank. Furthermore, Fig. 12 (b) shows the distribution of information among areas after applying the extension of the grid adjustment. From this result, we observe that the extension mitigates the nonuniformity of the geographical distribution.

The simulation procedure is described as follows:

1. Determine the cached data accessed (requested) by a client. This request is represented in the form of area (x, y, p), i.e., the client requests invalidation reports for several geographical data included in area (x, y, p). We set three area sizes (1, 2, and 4) to the value of p. The ratio of these sizes is $(p = 1 : p = 2 : p = 4) = (4 : 2 : 1)$. Furthermore, x and y are selected using the notion of a working set, which is a set of necessary areas consisting of the top 1,000 for the distribution of information per area. Furthermore, the working set model is used as an area locality, where 25% of all needs are issued from the working set.
2. Determine whether the requested data is updated according to update ratio γ. The definition of the update ratio is the ratio of the updated facilities' information that totals one per basic area.
3. The client receives the reports associated with the requested data according to the broadcast schedule.
4. This procedure is finished when all the reports has been completely received.



(a) Without grid adjustment.



(b) With grid adjustment.

Fig. 12 Distribution of information among areas.

This request procedure is repeated 10,000 times and the interval between any two consecutive requests is determined according to a Poisson distribution. One hash function is used and the storage size for cached data is infinite in this simulation.

If the bit length of the Bloom filter for the 1×1 filtering size is insufficient for the number of facilities' information, the number of requests for checking updates from the client to the server significantly increases, due to the frequent occurrence of the false positive. Therefore, we adjust the bit length to maintain the false positives at 1% or lower. According to the equation of the occurrence ratio of false positives, as described in Sect. 4.1, we need at least an 82-bit length if the update ratio of facilities' information is 5%.

As a metric for evaluating fragmentation, we employ

the number of repetitions of the start-up and the sleep-down until all the invalidation reports for the necessary area are completely received. We call this a fragment count. As for the access time, we use the average access period from the time when a client accesses caches until the time when it receives all of the necessary invalidation reports associated with the caches. Here, we define one time unit as the time necessary for broadcasting eight bits of a Bloom filter. As a concrete example of this time unit, we consider a data broadcasting service in Japan. ISDB-T (Terrestrial Integrated Services Digital Broadcasting) one-seg is a broadcasting service for mobile terminals such as cellular phones. In this service, the data rate is about 400 Kb/s; thus a time unit corresponds to 2×10^{-5} s.

In this simulation, we evaluate the three proposed methods FASM, MBSM, and MASM, in terms of the two performance metrics. In order to show the effectiveness of the proposed methods, we have to compare them with existing stateless broadcast-based cache-invalidation methods as described in Sect. 2. However, the existing methods consider only the average access time, whereas the proposed methods focus on not only the access time but also the fragmentation on broadcast schedule in such cache-invalidation protocols for geographical data. In addition, one of the objectives of this simulation is to confirm the balance between the average access time and fragmentation in the MBSM and MASM methods. Therefore, the performance of the FASM is used as a basic target for comparison to that of the other methods.

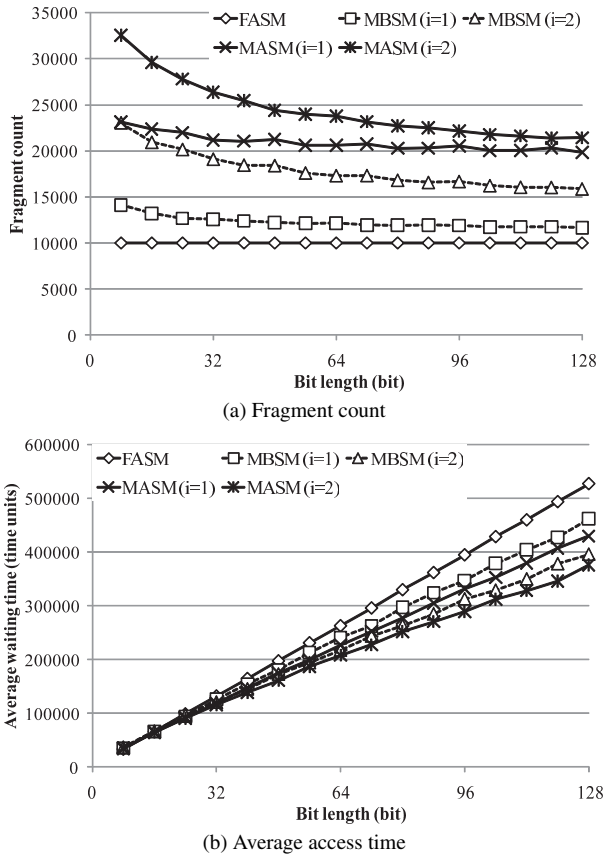


Fig. 13 Results for the FASM, MBSM, and MASM with grid adjustment (5% update ratio).

5.2 Results

5.2.1 Comparison of Proposed Schemes

First, we show the trade-off between fragmentation and the average access time, by means of the result that the MBSM and MASM can improve the average access time as compared with the FASM. In this experiment, the MBSM and MASM are evaluated in the cases of the half broadcast cycle ($i = 1$) and the quarter broadcast cycle ($i = 2$). We evaluate both performances of the three methods by varying the length of the Bloom filter. In addition, the proposed methods enable the extension of the grid adjustment.

Figure 13 (a) shows the result for the fragment count versus the bit length of a Bloom filter, when γ is equal to 5%. From Fig. 13 (a), we note that, for the MBSM and MASM methods, the fragment count decreases as the bit length increases. This is because the ratio of false positives is decreased by increasing the bit length of the Bloom filter. In particular, the proposed methods that use a large filtering size (that is, the MBSM and MASM) can improve the effectiveness of the increase in the bit length. As compared with the FASM, the fragment count of the MBSM ($i = 1$) increases by approximately 19%, but that of the MASM ($i = 1$) increases by approximately 103% in the case of the 88 bit length, which is enough to keep the false positive ratio at 1%.

Figure 13 (b) shows the result for the average access time versus the bit length of the Bloom filter when γ is 5%. In Fig. 13 (b), it is shown that the performance improvement for the MASM and MBSM becomes high as the bit length is longer, as compared with the FASM. This reason is similar to that in the result of the fragment count. Especially in the case of the 88-bit length, the MASM ($i = 2$) achieves the best performance, which is 25% better than the FASM. The MBSM ($i = 2$) also improves the access time by about 21%.

5.2.2 Impact of Update Ratio

Next, we examine the impact of the update ratio γ on the performance. In this experiment, we evaluate the fragment count and the average access time when γ is set to 10%. Table 1 (a) shows the fragment count for the MASM and MBSM in the case of the 88-bit length of Bloom filter. The improvement of the fragment count reduces by increasing γ, because the increase in the ratio of false positives by frequent updates prevents the client from detecting the updates in the large filtering size of the proposed methods. As compared with the performance when γ is 5%, the fragment count of the MBSM ($i = 2$) and MASM ($i = 2$) increases by about 14%.

Table 1 (b) shows the result for the average access time. As compared with the performance when γ is 5%, the aver-

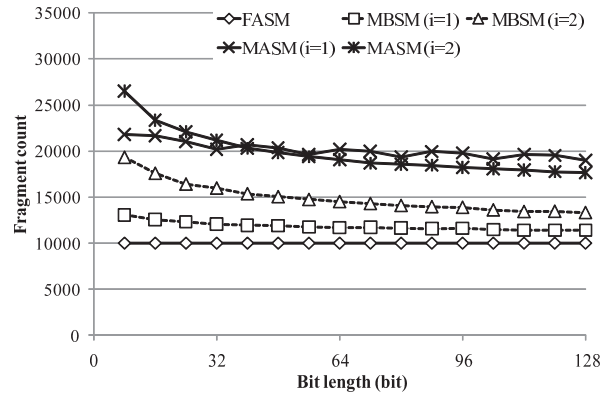
Table 1 Impact of update ratio γ on fragment count and average access time, in the case of the 88-bit length of Bloom filter.

(a) Fragment count ($\times 10^4$)

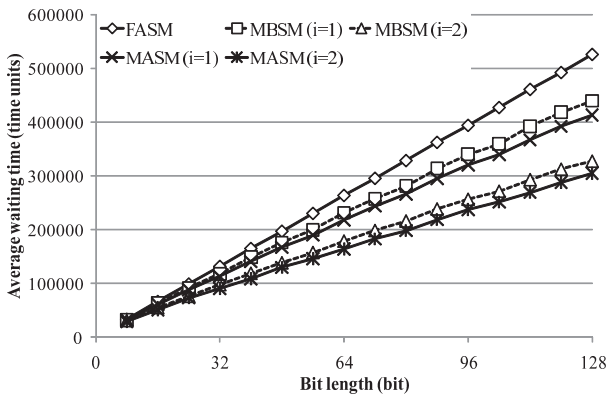
γ	MBSM (i = 1)	MBSM (i = 2)	MASM (i = 1)	MASM (i = 2)
5	1.19	1.66	2.03	2.25
10	1.26	1.89	2.11	2.56

(b) Average access time ($\times 10^5$ time units)

γ	MBSM (i = 1)	MBSM (i = 2)	MASM (i = 1)	MASM (i = 2)
5	3.24	2.84	3.05	2.70
10	3.60	3.30	3.30	3.16



(a) Fragment count



(b) Average access time

Fig. 14 Results for the FASM, MBSM, and MASM with grid adjustment and variable bit length (5% update ratio).

age access time of $\gamma = 10$ also increases. In particular, the increase for the MBSM ($i = 2$) and MASM ($i = 2$), which employ a large filtering size, is greater than the other protocols. This is because these protocols frequently wait for the next subcycles to avoid false positive, as the update ratio (i.e. invalid data) increases. In the case of the 88-bit length, the access time of the MBSM ($i = 2$) increases by about 16% and that of MASM ($i = 2$) increases by about 17%.

5.2.3 Effectiveness of Variable Length of Bloom filter

Next, we examine the effectiveness of the extension of the variable length of the Bloom filter. Figure 14 shows the result for the fragment count and the average access time for the proposed schemes. From Fig. 14 (a), we can observe that the fragment count for the MBSM ($i = 2$) and MASM ($i = 2$) can be significantly improved by enabling the extension, although the other proposed schemes slightly improve it. Furthermore, the same phenomenon is noted in the average access time. In particular, when the bit length is 88 bits, the MBSM ($i = 2$) can improve the average access time by 34%, while the increase in the fragment count is 40%, and the MASM ($i = 2$) reduces the average access time by 40% against an 85% increase in fragmentation, as compared with the FASM. Here, we provide actual access time in the case of the ISDB-T one-seg service. When the bit length is 88, the actual access time for FASM, MBSM ($i = 2$), and MASM ($i = 2$) is 7.3 s, 4.8 s, and 4.4 s, respectively,

Although the ratio of false positives is reduced by increasing the filtering size of the entire area, the total broadcast cycle increases. In the extension, the ratio can be reduced by assigning a long bit length only for a large filtering size, while restricting the increase in the total broadcast cycle. Therefore, the MBSM ($i = 2$) and MASM ($i = 2$), which use a large filtering size, can improve the effectiveness of the extension of the variable length of the Bloom filter. Thus, these methods can strike a balance between fragmentation and the average access time.

6. Conclusion

In this paper, we have proposed a broadcast-based cache-invalidation method for a geographical information service. We adopt the Bloom filter to express the invalidation report efficiently. Three methods, the FASM, MBSM, and MASM, are proposed for each purpose. The FASM aims to avoid fragmentation. The MBSM, which is the extended method of FASM, aims to reduce the access time with small fragmentation. The MASM aims to shorten the average access time. By computer simulation, we evaluate these three methods in terms of the fragment count and the average access time. As a result, compared with the FASM, the MBSM achieves an approximate 34% improvement in the average access time with a 40% increase in fragmentation. The MASM reduces the average access time by 40% against an 85% increase in fragmentation.

In the future, we will endeavor to evaluate the proposed methods in more detail, from the viewpoints of sensitivity of update ratio and scalability. Furthermore, we will address the technical issues occurring in such live environments.

Acknowledgement

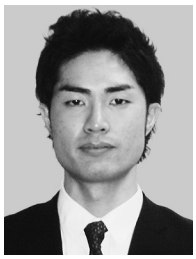
This work was partially supported by "The Kyushu University Research Superstar Program (SSP)", based on the budget of Kyushu University allocated under President's initiative and supported by KAKENHI (22300025 and 22700076).

References

[1] D. Barbara and T. Imielinski, "Sleepers and workaholics: Caching

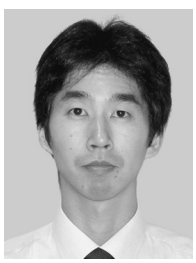
strategies in mobile environments," Proc. ACM SIGMOD Conf. on Management of Data, pp.1–12, 1994.
[2] Y. Huang, J. Cao, Z. Wang, B. Jin, and Y. Feng, "Achieving flexible cache consistency for pervasive Internet access," Proc. Fifth Annual IEEE Int'l Conf. on Pervasive Computing and Communications, PerCom '07, pp.239–250, 2007.
[3] S. Lim, W.-C. Lee, G. Cao, and C.R. Das, "Performance comparison of cache invalidation strategies for Internet-based mobile ad hoc networks," Proc. IEEE Int'l Conf. on Mobile Ad-hoc and Sensor Systems, pp.104–113, 2004.
[4] S. Lim, W.-C. Lee, G. Cao, and C.R. Das, "Cache invalidation strategies for Internet-based mobile ad hoc networks," Comput. Commun., vol.30, no.8, pp.1854–1869, 2007.
[5] Z. Wang, M. Kumar, S.K. Das, and H. Shen, "Dynamic cache consistency schemes for wireless cellular networks," IEEE Trans. Wirel. Commun., vol.5, no.2, pp.366–376, 2006.
[6] J. Cao, Y. Zhang, G. Cao, and L. Xie, "Data consistency for cooperative caching in mobile environments," Flagship Magazine of IEEE Computer Society, vol.40, no.4, pp.60–66, 2007.
[7] G. Cao, "A scalable low-latency cache invalidation strategy for mobile environments," IEEE Trans. Knowl. Data Eng., vol.15, no.5, pp.1251–1265, 2003.
[8] Y.-K. Chang, I.-W. Ting, and T.-H. Lin, "Dynamic cache invalidation scheme in IR-based wireless environments," Proc. 22nd Int'l Conf. on Advanced Information Networking and Applications, pp.697–704, 2008.
[9] A. Kahol, S. Khurana, S.K.S. Gupta, and P.K. Srimani, "A strategy to manage cache consistency in a distributed mobile wireless environment," IEEE Trans. Parallel Distrib. Syst., vol.12, no.7, pp.686–700, 2001.
[10] J. Jing, A.K. Elmagarmid, A. Helal, and R. Alonso, "Bit-sequences: A new cache invalidation method in mobile environments," ACM/Baltzer J. Mobile Networks and Applications (MONET), vol.2, no.2, pp.115–127, 1997.
[11] H. Safa, H. Artail, and M. Nahhas, "A cache invalidation strategy for mobile networks," J. Network and Computer Applications, vol.33, no.2, pp.168–182, 2010.
[12] D.H. Alassi and R. Alhajj, "Enhanced scalable asynchronous cache consistency scheme for mobile environments," University of Calgary, Technical Report 2011-989-01, 2011.
[13] J. Xu, X. Tang, D.L. Lee, and Q.L. Hu, "Cache coherency in location-dependent information services for mobile environment," Proc. 1st Int'l Conf. on Mobile Data Management, pp.182–193, 1999.
[14] K.C.K. Lee, W.-C. Lee, H.V. Leong, B. Unger, and B. Zheng, "Efficient valid scope for location-dependent spatial queries in mobile environments," J. Sofware, vol.5, no.2, pp.133–145, 2010.
[15] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," Commun. ACM, vol.13, no.7, pp.422–426, 1970.
[16] Y. Yang, R. Dunloap, M. Rexroad, and B.F. Cooper, "Performance of full text search in structured and unstructured peer-to-peer systems," Proc. Int'l Conf. on IEEE INFOCOM, 2006.
[17] P. Reynolds and A. Valdet, "Efficient peer-to-peer keyword searching," Proc. International Middleware Conference, 2003.
[18] Kiwi-W Consortium. Input for ISO Physical Storage Format, 28 March 2000. http://www.kiwi-w.org/

Shigeaki Tagashira received the B.Eng. degree from Ryukoku University in 1996; and the M.Eng. and D.Eng. degrees in information science from Nara Institute of Science and Technology (NAIST) in 1998 and 2000, respectively. He worked as a research associate at Hiroshima University from 2000 to 2007. Since 2007 he has been a project associate professor at Kyushu University. His current research interests include ubiquitous computing and system software. He is a member of the Information Processing Society of Japan (IPSJ) and Institute of Electrical and Electronics Engineers (IEEE).



Yutaka Kaminishi received the B.Eng. degree from Kyushu Institute of Technology in 2007; and M.Eng. degree in computer science from Kyushu University, Japan, in 2009. His current research interests include embedded system and mobile computing.

Yutaka Arakawa received B.E., M.E., and Ph.D., from Keio University, Japan, in 2001, 2003, and 2006, respectively. Since 2001, he has researched about optical network architecture and traffic engineering, especially in Optical Burst Switching. From 2004 to 2006, he was a research assistant of Keio University COE Program in "Optical and Electronic Device on Access Network" by Ministry of Education, Culture, Sports, Science and Technology, Japan. He was an assistant professor of Yamanaka lab. in Keio University from 2006 to 2008. He is now assistant professor of Fukuda and Nakanishi lab. in Kyushu University, and is mainly engaged in research on network applications. He received Best Paper Award of APCC/COIN2008, Best Paper Award of SIG-MBL2009, Excellent paper award of DICOMO2010, Best Presentation Award of DICOMO2010. He is a member of the Institute of Electrical and Electronics Engineers (IEEE) of USA, and the Information Processing Society of Japan (IPSJ).



Teruaki Kitasuka received his B.E. Degree from Kyoto University, M.E. Degree from Nara Institute of Science and Technology (NAIST), and D.E. Degree from Kyushu University, Japan, in 1993, 1995 and 2006, respectively. He worked for SHARP Corporation from 1995 to 2001. He was a research associate in the Faculty of Information Science and Electrical Engineering, Kyushu University, from 2001 to 2007. Since 2007, he has been an associate professor in the Graduate School of Science and Technology, Kumamoto University. His research interests include location-aware systems, and ubiquitous, mobile and embedded computing. He is a member of IEEE, IPSJ and DBSJ.

Akira Fukuda received the B.Eng., M.Eng., and Ph.D. degrees in computer science and communication engineering from Kyushu University, Japan, in 1977, 1979, and 1985, respectively. From 1977 to 1981, he worked for the Nippon Telegraph and Telephone Corporation, where he engaged in research on performance evaluation of computer systems and the queueing theory. From 1981 to 1991 and from 1991 to 1993, he worked for the Department of Information Systems and the Department of Computer Science and Communication Engineering, Kyushu University, Japan, respectively. In 1994, he joined Nara Institute of Science and Technology, Japan, as a professor. Since 2001 and 2008, he has been a professor of Graduate School of Information Science and Electrical Engineering, and director of System LSI Research Center, Kyushu Univ., Japan, respectively. His research interests include embedded systems, ubiquitous computing, system software (operating systems, compiler, and run-time systems), parallel and distributed systems, and performance evaluation. He received 1990 IPSJ (Information Processing Society of Japan) Research Award and 1993 IPSJ Best Author Award. He is a member of the ACM, the IEEE Computer Society, the IPSJ, and the Operations Research Society of Japan.