On Signature Caching of Wireless Broadcast and Filtering Services

Dik Lun Lee
Department of Computer Science
University of Science and Technology
Clear Water Bay, Hong Kong
dlee@cs.ust.hk
FAX: (852) 2358-1477

Wang-Chien Lee
Dept of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210-1277, USA
wlee@cis.ohio-state.edu
FAX: 614-292-2911

Abstract

This paper discusses signature caching strategies to reduce power consumption for wireless broadcast and filtering services. Four caching policies are compared in terms of tune-in time and access time. With reasonable access time delay, all of the caching policies help in reducing tune-in time for the two-level signature scheme. Moreover, two cache replacement policies are presented and compared by simulation. The result shows that, when the cache size is small, caching only the integrated signatures is recommended. When the size of cache is greater than that of the integrated signatures, caching both of the integrated and simple signatures is better.

1 Introduction

Recent development in wireless communication and personal computing techniques have opened up new opportunities for mobile computing. In the future, numerous battery powered palmtop computers will be able to receive various kinds of services over wireless communication channels [6]. Information broadcast will be an important information dissemination method for mobile applications, because it can scale up to an arbitrary number of mobile users. Thus, *broadcast and filtering services*, e.g., stock quote, weather and traffic information, and indoor sales aid [7], will be one of the most important mobile applications.

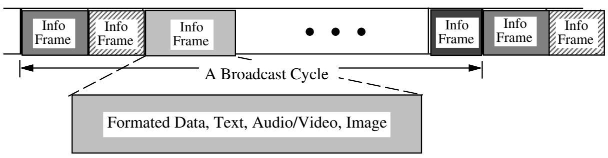

Figure 1: Information Stream.

Broadcast and filtering services consist of two ends. At the broadcast end, the mobile support station sends out a series of *information frames.* (See Figure 1.) An information frame is a logical unit of information broadcast on the air and may consist of multimedia information, including text, image, audio/video and other related data. Frames may vary in size; they consist of *packets* which are the physical units of broadcast. A frame contains a header (not shown in the figure) for synchronization as well as meta-information indicating the type and length of the frame. At the filtering end, users are allowed to specify conditions on the frames they are interested in. The mobile computers will receive information frames over the wireless channel, filter out unwanted information, and only present to the users frames matching the specified conditions. Since the information frames are periodically broadcast, a complete broadcast of the information frames is called a *broadcast cycle*. From the user's viewpoint, the broadcast information is perceived as a stream of frames flowing along the time axis. Logically, there is no specific start and end frames for a broadcast cycle; a broadcast cycle starts with any frame and ends when the frame appears again. In a broadcast cycle, some important information frames may be replicated (i.e., frames with the same contents but treated as different frames). Information frames may be inserted, deleted, and modified. The updates will be reflected in the subsequent broadcast cycles.

In order to make mobile computers portable, small batteries, such as AA or AAA batteries, are used as the main power source. These batteries have small capacity and need recharging or replacement after a short period of usage. Therefore, power conservation is an important issue for applications on mobile computing environment. A well-designed mobile software has to take power consumption into consideration in addition to execution time.

There are two factors affecting power consumption in mobile computers: (1) Mobile computers can be switched between *active* mode (full power) and *doze* mode [5], and (2) receiving messages consumes less energy than sending messages. By switching between active and doze modes, power consumption is reduced.

Tune-in time and *access time* are two of the merits for performance evaluation. The duration that a mobile computer must stay in active mode to answer a query is called the tune-in time, which is proportional to power consumption. Access time is the time required to collect all qualified frames. Without any access aid, both the tune-in time and access time are equal to the length of the broadcast cycle, because it is necessary to scan through all of the frames in a broadcast cycle to pick up the qualified frames. This is very inefficient in power consumption, because typically only a few frames in a cycle satisfy the user request.

Access methods can be developed so that the mobile computer can be turned off when the frame being broadcast is not qualified [8,9]. In order to tell which frames would qualify ahead of time, auxiliary information about the contents of the frames must be added. In a previous paper, we proposed signature schemes to index the information frames on the air [10]. With reasonable access time delay, the proposed signature indexing methods dramatically reduce the tune-in time. In this paper, we address the caching policies for maintaining signatures in the main memory of the mobile computers to further reduce tune-in time.

The rest of the paper is organized as follows. In Section 2, we briefly review a two-level signature scheme for information broadcast and filtering services. In Section 3, we discuss caching policies for the two-level signature scheme. In Section 4 and Section 5, we analyze and compare the caching policies, respectively. In Section 6, we propose two cache replacement policies and compare their cache hits by simulation. Finally, Section 7 concludes the paper.

2 Two-level Signature Indexing

A signature is basically an abstraction of the information stored in an information frame. They are constructed from and broadcast together with the information frames. A number of methods for generating signatures have been proposed in the context of database and document retrieval [3,4]. Since the signature file technique has been discussed extensively in the literature, we won't repeat it here. Readers are referred to [3] and [4] for details.

The signature of an information frame is formed by first hashing each value in the frame into a bit string and then superimposing together all bit strings generated from the frame into the frame signature. During filtering, a query signature is constructed in the same way and then compared to the frame signatures. If the result is negative, the information frame can be skipped. Otherwise, the information frame must be further compared with the query to distinguish *false drops* (i.e., frames which don't qualify the query even though their signatures match the query signature) from *true matches* (i.e., frames which truly qualify the query). The signature technique is very suitable for filtering information frames in a wireless broadcasting environment, because it offers the following advantages:

- Signature techniques may be generally applied to various types of information media.
- Signature techniques are particular good for multi-attribute retrieval, which is necessary for specifying precise filtering conditions.
- Signatures are very easy to generate and search; thus, they are suitable for mobile computers where realtime searching with limited buffer space is required.
- A signature is very short compared to an information frame; therefore, the access time won't be increased drastically.
- A signature file is basically a sequential file structure. This makes it easy to "linearize" the signature file for broadcasting on air and scanning by a mobile computer.

In the paper, the caching policies are based on the *two-level signature* scheme for wireless broadcast and filtering services [10]. A similar scheme has been discussed for partial match retrieval [11]. As the name suggested, the two-level signature scheme consists of two types of signatures: the *simple signature* and the *integrated signatures*. A simple signature is constructed for each information frame as described above. An integrated signature is constructed from a group of consecutive frames, called a *frame group*. In the two-level signature scheme, the upper level signatures are integrated signatures and the lowest level signatures are simple signatures.

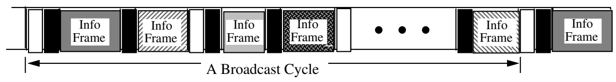

Figure 2: Two-level Signature Scheme.

Figure 2 illustrates the two-level signature scheme. The white signatures in the figure are integrated signatures. An integrated signature indexes all of the information frames between itself and the next integrated signature. (In the figure, an integrated signature indexes two information frames.) The black signatures are simple signatures for the corresponding information frames. To reduce the chance of getting false drops, the hashing functions used in generating the integrated signatures and simple signatures are different.

To answer a query, a query signature is generated for each level of the signatures. After tuning into the broadcast channel, the corresponding query signature is used to compare with different levels of signatures. If a signature fails in the comparison, the mobile computer switches to doze mode until a signature at the same or upper levels arrives. Otherwise, the mobile computer stays in active mode and continues the filtering process. Consider Figure 2 as an example. Query signatures S_Q and $S_Q{}'$ are constructed for the integrated and simple signature levels, respectively. When an integrated signature is received, S_Q is used to match with the signature. If the match fails, the mobile computer will go into doze mode until the next integrated signature arrives. If the match is successful, $S_Q{}'$ is used to match with the following simple signature. If they match, the corresponding information frame is received; if not, the mobile computer may go into doze mode until the next signature arrives.

3 Signature Caching

To reduce battery power consumption for mobile computers, we have to reduce the amount of information down loaded from the communication channel. Caching of frequently accessed information in the mobile computers can reduce tune-in time considerably, and thus power consumption, because information can be fetched from the cache without tuning into the communication channel. Due to physical constraints, mobile computers usually have relatively small memory. Thus, caching large chunks of multi-media information frames is infeasible. Compared to the information frames, signatures need a rather small storage overhead and they contain critical information to support various kinds of queries. Thus, they are very suitable as the caching entity.

Previous studies [1,2] have discussed cache management policies and invalidation strategies for information dissemination based on broadcast. In [2], the effect of disconnections on three cache invalidation approaches are discussed, while [1] proposes to use wireless channels as broadcast disks and discuss the associated prefetch and caching strategies. In our study, we consider the signature caching schemes for information broadcast and filtering services. Two-level signature scheme is used in our discussion. Two factors affecting signature maintenance in cache are considered:

Invalidation notice: An invalidation notice indicates which cached signatures are stale.

In order to support information filtering, the signatures in the cache have to be accurate. Therefore, the information server has to provide invalidation information to mobile computers. If some information frames and their corresponding signatures are updated, the information server will indicate the changes in the invalidation notice embedded in the broadcast. There are two kinds of invalidation notices, aiming at different situations.

• Bit tags: For each signature, a 1-bit tag is allocated to indicate whether the corresponding signature has been changed since the last broadcast cycle. The overhead of the bit tags is very low. However, the invalidation information only indicates changes with respect to the immediate preceding cycle. If a mobile computer has stopped tracking the invalidation information over a period of time, it cannot tell if the signatures in the cache is valid or not, even though the invalidation notice indicates no changes. Therefore, when a mobile client tunes into a channel, it has to reload the signatures into its cache memory in accordance with the caching policies used.

• Version numbers: To reduce the cost of reloading the signatures into the cache every time the mobile computer tunes into a channel, multiple-bit version numbers may be used to serve as the invalidation notice. In the broadcast, a version number is assigned to each signature. If the signature is modified, its version number is incremented by 1 (and reset to 0 when it reaches the largest number representable).

The size of the version number is dependent on the frequency of updates on the signatures. However, it should not exceed that of the simple signatures, because, if it does, it will cost the mobile clients less by simply reloading all simple signatures. Due to the limited length allowed for the version numbers, the information server has to determine the period when the same version number won't appear twice. The mobile clients will assume that the signatures they cached expire after they lost track of the channel for the specific period of time and reload the signatures in accordance with the caching policy used. If the mobile users listen to the broadcast channels before the version numbers expired, they only need to reload the signatures which are changed during the off period.

Refresh strategy: The refresh strategy determines which signatures to maintain in the cache.

• Active refresh: Active refresh maintains all of the signatures in the cache. In order to maintain the accuracy of the signatures, a mobile computer has to load the updated signatures into the cache based on the invalidation notices. As a result, the performance of the active refresh strategy is influenced by the number of updates on the information frames.

• Passive refresh: Passive refresh only keeps the previously accessed signatures in the cache. Instead of maintaining all of the signatures in the cache, only the signatures received in the previous filtering process are kept in the cache. According to the invalidation notices, invalid signatures are cleared from the cache without refresh. Therefore, the performance of information filtering using passive refresh caching policies is not affected by the number of updates on the information frames.

4 Analysis of Signature Caching Policies

In the following, we propose four signature caching policies based on the invalidation notice and cache refresh strategies. The cost models for tune-in time and access time of the policies are derived and their performance is compared. We assume that a two-level signature scheme is used in information broadcasting and filtering. The caching policies may be easily applied to other signature schemes.

4.1 Symbols and Parameters

- A: number of information frames in a broadcast cycle.
- I: number of integrated signatures in a broadcast cycle.
- P_f^s: false drop probability for simple signatures.
- P_f^i: false drop probability for integrated signatures.
- P_s: selectivity of a query.
- P_u: probability of updates on simple signatures.

k: the number of information frames indexed by an integrated signature.
l: locality of true matches (average number of true matches in a frame group).
m: length of a signature in bits.
n: the average number of packets in an information frame.
p: the number of bits in a packet.
r: the number of packets in a signature ($r = \lceil m/p \rceil$).
s: the number of bit strings which are superimposed into a signature.
t: the average number of cycles a mobile computer is disconnected from a channel.
u: locality of updated simple signatures in a frame group.
v: size of version numbers in bits.

4.2 Two-level Signature Scheme

In the following, we show the cost models for tune-in time and access time estimation of the two-level signature scheme [10]. The length of a complete cycle is:

$$CYCLE = DATA + SIG_i + SIG_s,$$

where SIG_i and SIG_s are the number of packets for simple and integrated signatures, respectively, and $DATA$ is the total size of the information frames in packets. The average initial probe time, the period from the user first tune into a channel to the arrival of a signature, is:

$$PROBE = \frac{k \cdot n^2 + (k+1)r^2 + 2 \cdot k \cdot nr}{2(k \cdot n + (k+1)r)}.$$

Thus, the access time is:

$$ACCESS = PROBE + CYCLE.$$

The tune-in time in the initial probe period is:

$$PT = \frac{k \cdot n^2 + (k+1)r^2}{2(k \cdot n + (k+1)r)}.$$

The total tune-in time can be approximated as follows.

$$\begin{aligned} TUNE = {} & PT + SIG_i \\ & +(I - \lceil P_s \cdot A/l \rceil) \cdot P_f^i \cdot (k \cdot r + k \cdot P_f^s \cdot n) \\ & +\lceil P_s \cdot A/l \rceil (k \cdot r + l \cdot n + (k-l) \cdot P_f^s \cdot n). \end{aligned}$$

4.3 Bit Tags and Active Refresh (BA)

The BA policy maintains all of the integrated signatures and simple signatures in the cache memory. For each frame group, we use bit tags in front of the integrated signature to indicate the update status of the signatures generated from the group. When a bit tag signals a change, the corresponding signature has to be loaded into the cache memory. The query signatures are then matched with the signatures maintained in the cache to decide which information frames may be skipped and which have to be brought into the mobile computers for false drop elimination. $k+1$ bits are necessary for a group of k frames. The total broadcast overhead for invalidation information is:

$$II_{BA} = \lceil (k+1)/p \rceil \cdot \lceil A/k \rceil.$$

Note that the invalidation information may share a packet with an integrated signature to make more compact use of the packets. Here we estimate its upper bound overhead.

The access time for the caching policies is[1]:

$$ACCESS = PROBE + II + CYCLE.$$

For new users or those who lost track of the invalidation information, the tune-in time for loading signatures into the cache in the initial cycle is the number of packets for bit tags and signatures in a cycle:

$$Init_{BA} = II_{BA} + SIG_i + SIG_s.$$

Since the cache loading process may be combined with the filtering process, a query may be answered while the signatures are being loaded into the cache. Therefore, the actual tune-in time is the sum of the initial probe time, the signature loading time, and the time to load the qualified information frames for false drop verification. Thus, the initial tune-in time is:

$$\begin{aligned} Tune_{BA}^{init} = {} & PT + Init_{BA} + (I - \lceil P_s \cdot A/l \rceil) \cdot P_f^i \cdot k \cdot P_f^s \cdot n \\ & +\lceil P_s \cdot A/l \rceil (l \cdot n + (k-l) \cdot P_f^s \cdot n). \end{aligned}$$

For the subsequent queries, the tune-in time is consumed for loading the modified signatures and listening to information frames for false drop elimination. Since the update probability of the simple signatures is very close to that of the information frames, the average numbers of the modified simple and integrated signatures are $P_u \cdot A$ and $\lceil P_u \cdot A/u \rceil$ respectively. The tune-in time for the subsequent queries is:

$$\begin{aligned} Tune_{BA}^{next} = {} & PT + II_{BA} + P_u \cdot A \cdot r + \lceil P_u \cdot A/u \rceil \cdot r \\ & +(I - \lceil P_s \cdot A/l \rceil) \cdot P_f^i \cdot k \cdot P_f^s \cdot n \\ & +\lceil P_s \cdot A/l \rceil (l \cdot n + (k-l) \cdot P_f^s \cdot n). \end{aligned}$$

4.4 Version Numbers and Active Refresh (VA)

In this policy, version numbers for the signatures generated from a frame group are broadcast before the integrated signature. If a broadcast version number corresponding to a signature is different from that of the same signature in the cache memory, the signature in the cache is not valid. Therefore, the updated signature has to be brought into the cache. Due to the constraint on the size of the version numbers, the version numbers

[1] Since the formulae for access time is the same as above for the other policies, we will not repeat it in the following sections.

are valid only for a certain period of time. If a mobile computer loses track of the broadcast channel over that period of time, we will consider it as a new client for the channel. A new client has no prior knowledge of the channel and thus has to load the signatures into its cache in accordance with the caching policy. For an old client, however, only outdated signatures will be deleted and reloaded.

Assume that the size of a version number is v bits. The period of time in which the version number is guaranteed valid is $2^v - 1$ broadcast cycles.
The total overhead for the invalidation information is:

$$II_{VA} = \lceil (k+1) \cdot v/p \rceil \cdot \lceil A/k \rceil.$$

For a new user to load signatures into the cache, the tune-in time is:

$$Init_{VA}^{new} = II_{VA} + SIG_i + SIG_s.$$

For an old user who lost track of the broadcast channel for t cycles, where $t < 2^v - 1$, the average number of simple signatures which have not been changed during this period of time is $A \cdot (1 - P_u)^t$. Therefore, the number of simple signatures which have to be reloaded is $A - A \cdot (1 - P_u)^t$. Similarly, the number of integrated signatures which have to be reloaded is $I - I \cdot (1 - \frac{\lceil A \cdot P_u/u \rceil}{I})^t$. Therefore, the signature loading time for an old user is:

$$\begin{aligned} Init_{VA}^{old} &= II_{VA} + (A - A \cdot (1 - P_u)^t) \cdot r \\ &\quad + (I - I \cdot (1 - \frac{\lceil A \cdot P_u/u \rceil}{I})^t) \cdot r \end{aligned}$$

If the initial cache loading process of a client is combined with the first query evaluation, the total tune-in time is:

$$\begin{aligned} Tune_{VA}^{init} = {} & PT + Init_{VA}^{*} + (I - \lceil P_s \cdot A/l \rceil) \cdot P_f^i \cdot k \cdot P_f^s \cdot n \\ & + \lceil P_s \cdot A/l \rceil (l \cdot n + (k - l) \cdot P_f^s \cdot n), \end{aligned}$$

where $Init_{VA}^{*}$ is either $Init_{VA}^{new}$ or $Init_{VA}^{old}$ depending on whether the user is new to the channel or not.

For the subsequent queries, the tune-in time is consumed for loading the modified signatures and listening to information frames for false drop elimination. The tune-in time for the subsequent queries is:

$$\begin{aligned} Tune_{VA}^{next} &= PT + II_{VA} + P_u \cdot A \cdot r + \lceil P_u \cdot A/u \rceil \cdot r \\ &\quad + (I - \lceil P_s \cdot A/l \rceil) \cdot P_f^i \cdot k \cdot P_f^s \cdot n \\ &\quad + \lceil P_s \cdot A/l \rceil (l \cdot n + (k - l) \cdot P_f^s \cdot n). \end{aligned}$$

4.5 Bit Tags and Passive Refresh (BP)

In the BP scheme, only the signatures received in previous queries are kept in the memory. Instead of actively updating all of the signatures in the cache memory, the modified signatures are deleted from the cache in accordance with the bit tags. New signatures are brought into the cache only if they are needed in the current filtering process. The bit tags corresponding to signatures of a frame group are broadcast before the integrated signature. If the bit tags show no change of the signatures in the frame group, it will use the integrated signature in cache for processing the query. If the bit tags indicate changes, the corresponding integrated and simple signatures in cache are deleted. If the integrated signature is updated, it will be loaded into the cache. The integrated query signature is then compared with the integrated signature in cache. If the comparison between the query signature and the integrated signature is a match, the simple signatures not residing in the cache will be loaded into the cache and compared to the simple query signature. If the comparison with the integrated signature fails, the mobile computer simply tunes off without refilling the cache for the deleted simple signatures.

The overhead for the bit tags is:

$$II_{BP} = \lceil (k+1)/p \rceil \cdot \lceil A/k \rceil.$$

Because of the passive manner on signature caching adopted by the BP policy, we don't consider the tune-in time needed for loading signatures. Since we don't bring all of the signatures into the cache, the tune-in time for the initial query is the same as answering a query without caching plus the overhead for the bit tags.

$$\begin{aligned} Tune_{BP}^{init} &= PT + II_{BP} + SIG_i \\ &\quad + (I - \lceil P_s \cdot A/l \rceil) \cdot (P_f^i \cdot k \cdot r + P_f^i \cdot k \cdot P_f^s \cdot n) \\ &\quad + \lceil P_s \cdot A/l \rceil (k \cdot r + l \cdot n + (k - l) \cdot P_f^s \cdot n). \end{aligned}$$

For each query, the average number of simple signatures used in signature comparison is:

$$A_{comp} = (I - \lceil P_s \cdot A/l \rceil) \cdot P_f^i \cdot k + \lceil P_s \cdot A/l \rceil \cdot k.$$

Let I_{cache} and A_{cache} be the number of integrated and simple signatures in cache, respectively. For the two-level signature scheme, every integrated signature is checked to decide whether the corresponding simple signatures and frame groups may be skipped. Thus, $I_{cache} = I$. The number of integrated signatures deleted from and brought into the cache is the same as the number of integrated signatures modified: $\lceil P_u \cdot A/u \rceil$. The average number of simple signatures purged from the cache is the number of simple signatures in cache times the percentage of updates, i.e., $P_u \cdot A_{cache}$. The simple signatures brought into the cache are those corresponding to successful matches between the integrated signatures and integrated query signature but not residing in the cache: $A_{comp} \cdot (1 - \frac{A_{cache}}{A} \cdot (1 - P_u))$.

For the subsequent queries, the tune-in time is:

$$\begin{aligned} Tune_{BP}^{next} = {} & PT + II_{BP} + \lceil P_u \cdot A/u \rceil \cdot r \\ & + A_{comp} \cdot (1 - \frac{A_{cache}}{A} \cdot (1 - P_u)) \cdot r \\ & + (I - \lceil P_s \cdot A/l \rceil) \cdot P_f^i \cdot k \cdot P_f^s \cdot n \\ & + \lceil P_s \cdot A/l \rceil (l \cdot n + (k - l) \cdot P_f^s \cdot n). \end{aligned}$$

4.6 Version Numbers and Passive Refresh (VP)

Assume that the size of a version number is v bits. The total broadcast overhead for the invalidation information is:

$$II_{VP} = \lceil (k+1) \cdot v/p \rceil \cdot \lceil A/k \rceil.$$

Similar to the BP policy, new signatures are brought into the cache only when they are needed in the current filtering process. The tune-in time for the initial query is the same as answering a query without caching plus the overhead for version numbers. However, depending on whether the mobile computer is new to the channel, the tune-in time for the initial query will be different. For a new client, the initial tune-in time is:

$$\begin{aligned} Tune_{VP}^{init,new} &= PT + II_{VP} + SIG_i \\ &+ (I - \lceil P_s \cdot A/l \rceil) \cdot (P_f^i \cdot k \cdot r + P_f^i \cdot k \cdot P_f^s \cdot n) \\ &+ \lceil P_s \cdot A/l \rceil (k \cdot r + l \cdot n + (k-l) \cdot P_f^s \cdot n). \end{aligned}$$

Assume that the number of integrated and simple signatures cached in a mobile client before it tunes off is I_{cache} and A_{cache}, respectively. In the two-level signature scheme, since every integrated signature is examined, a complete set of the integrated signatures has to be kept in the cache. Therefore, $I_{cache} = I$. For an old client which has been tuned off for t cycles, the number of valid simple signatures in cache is $A_{cache} \cdot (1 - P_u)^t$ and the number of valid integrated signatures in cache is $I \cdot (1 - \frac{\lceil A \cdot P_u/u \rceil}{I})^t$. As a result, the tune-in time for an old user's initial query is:

$$\begin{aligned} Tune_{VP}^{init,old} &= PT + II_{VP} + (I - I(1 - \frac{\lceil A \cdot P_u/u \rceil}{I})^t) \cdot r \\ &+ A_{comp} \cdot (1 - \frac{A_{cache}}{A}(1 - P_u)^t) \cdot r \\ &+ (I - \lceil P_s \cdot A/l \rceil) \cdot P_f^i \cdot k \cdot P_f^s \cdot n \\ &+ \lceil P_s \cdot A/l \rceil (l \cdot n + (k-l) \cdot P_f^s \cdot n). \end{aligned}$$

After the initial cycle, there is no difference between new and old clients. Therefore, the average tune-in time for the subsequent queries is:

$$\begin{aligned} Tune_{VP}^{next} &= PT + II_{VP} + \lceil P_u \cdot A/u \rceil \cdot r \\ &+ A_{comp} \cdot (1 - \frac{A_{cache}}{A} \cdot (1 - P_u)) \cdot r \\ &+ (I - \lceil P_s \cdot A/l \rceil) \cdot P_f^i \cdot k \cdot P_f^s \cdot n \\ &+ \lceil P_s \cdot A/l \rceil (l \cdot n + (k-l) \cdot P_f^s \cdot n). \end{aligned}$$

5 Comparison of Signature Caching Policies

There are several factors, such as the percentage of updates, size of signatures, and number of information frames in a group, affecting the tune-in time of a caching policy. In the following, we vary these factors to observe their influence on the caching policies.

Table 1: Parameters of the cost models.

$k = 4$	$l = 1$	$n = 1000$	$p = 128$	$s = 100$
$u = 1$	$v = 10$	$A = 10000$	$P_s = 0.01$	$P_u = 0.01$

First, we compare the tune-in time of the caching policies for the first eight queries after a mobile user connects to a broadcast channel. Important parameter values used in the comparison are listed in Table 1. We fix the signature size to 10 packets and the average percentage of frames updated in each cycle to 1%. The locality of the modified frames in a group is 1. Based on the formulae developed in [10], the false drop probabilities for integrated signatures and simple signatures, i.e., P_f^i and P_f^s, can be derived.

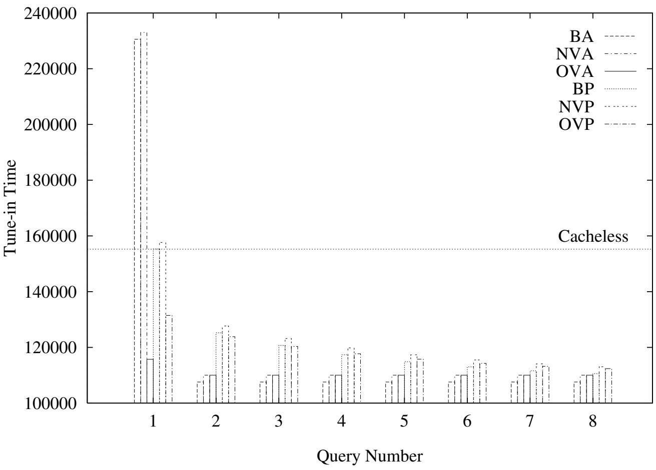

Figure 3: Tune-in time vs query number.

We use the tune-in time of a cacheless two-level scheme as reference. In Figure 3, the symbols for different policies are self-explanatory except that we add N and O in front of VA and VP to distinguish the cases for new and old users. New users have no signatures in cache while old users have some signatures left in the cache before they were switched off. For the old users, we assume that they have disconnected for four broadcast cycles. Since the bit tag methods use only one bit per frame, it would be unfair to assume that the bit tags occupy an entire packet. Therefore, we combine the bit tags with the integrated signatures in the evaluation. On the other hand, the length of a version number is set to 10 bits.

Figure 3 shows the tune-in time of the policies for the initial and subsequent queries. In general, the tune-in time of the initial query is much higher than that of the subsequent queries, because signatures have to be loaded into the caches in the initial query. Furthermore, only the BA and NVA policies have dramatically higher tune-in time than the cacheless two-level scheme, because they have to load all of the integrated and simple signatures into the cache. OVA and OVP have the best tune-in time, because they have signatures left in the cache from the previous connection. OVA is better than OVP, because OVA has more signatures left in the cache. On the other hand, the tune-in time of the passive policies for new users is close to the cacheless scheme, because the passive policies are dependent on the signature filtering process.

For the subsequent queries, the BA policy has the best performance, while NVA and OVA, which have identical tune-in time, are closely behind it. The difference between the BA and VA policies is due to the overhead of the invalidation information. In this comparison, the active policies outperform the passive policies. However, for subsequent queries, the passive policies gradually catch up. Also note that, the update rate is set to 1%, which is rather low. As we will show later, the update rate plays an important role in the difference between the active and passive policies.

From the data, all of the caching policies have better average tune-in time than the cacheless method after only listening to the channel for three broadcast cycles. Thus, we conclude that the signature schemes in general will benefit from the caching techniques.

Next, we compare the tune-in time of the caching policies by changing the size of the signatures. For this comparison, we calculate the average tune-in time for the initial query and two subsequent queries. To simplify our comparison, we use the average tune-in time for old users in the version number policies.

Figure 4 shows that VA < VP < BP < BA. In the figure, the tune-in time of BA is higher than that of the cacheless approach as the signature size increases. In practice, however, the signature size is chosen to yield the best performance of a caching policy. Therefore, the comparison should emphasize on the minimal points (marked in the figure) of the policies.

The above comparison shows the best performance of the policies. However, it didn't show their overhead on access time. In Figure 5, we show the average tune-in time of the first three queries against the access time for the proposed caching policies. For the cacheless, index-free filtering method, the tune-in time and access time are equal to the broadcast cycle, i.e., 10,000,000 packets with the parameters we set. As shown in the figure, the tune-in time for all of the cacheless and cached signature filtering methods rapidly decrease until the access time delay is at around 150,000 packets. We also observe that the cached signature filtering methods have better tune-in time than the cacheless method does when they are allowed to have more than 75,000 packets of access time delay. Also note that, the tune-in time performance of the cached signature filtering methods is comparable to the cacheless signature scheme when the access time delay is less than 75,000 packets. As the number of subsequent queries increases, the tune-in time for the caching policies will be lower. If enough subsequent queries are included, the caching policies will be absolutely better than the cacheless method (i.e., no cross-over point).

In addition to the factors discussed above, the number of information frames grouped together to generate an integrated signature also affects tune-in time performance in two ways. On one hand, increasing the number of frames grouped together allows more compact storage of the invalidation information in packets and thus reduces its overhead; on the other hand, the information abstracted and stored into the integrated signatures will increase, thus resulting in higher false drop rates. Figure 6 compares the caching methods based on the group size. In order to more easily observe the impact of the group size on fitting invalidation information into packets, we don't combine the bit tags with integrated signatures as we did in the previous comparisons. Also, we use a packet size of 4 bytes and a signatures size of 32 packets. In the figure, as the group size increases, we find that the tune-in time of the bit tag policies drops initially but becomes stable later. The initial drop is due to the more compact storage of the bit tags and the reduced number of integrated signatures. However, the factor of false drops dominates later on. The version number policies perform better mainly because they have inherited cached signatures from previous connections.

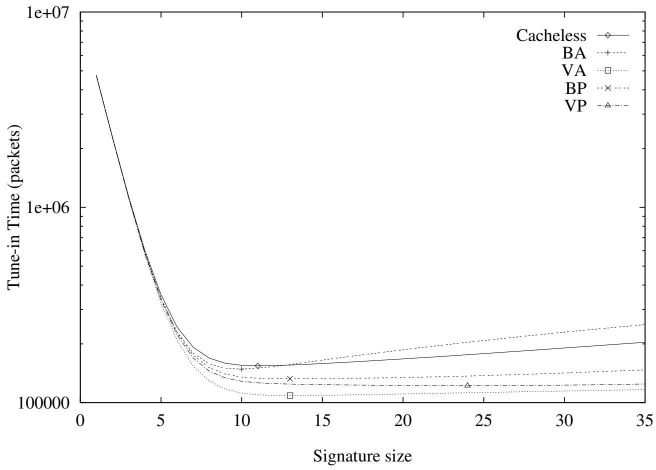

Figure 4: Tune-in time vs signature size.

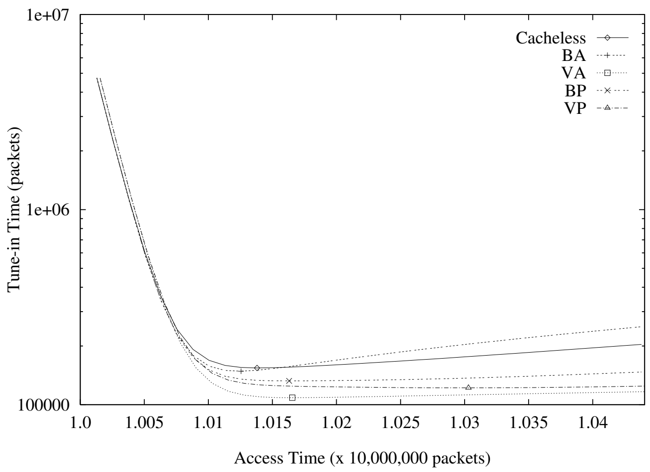

Figure 5: Tune-in time vs access time.

Finally, we increase the percentage of updated frames

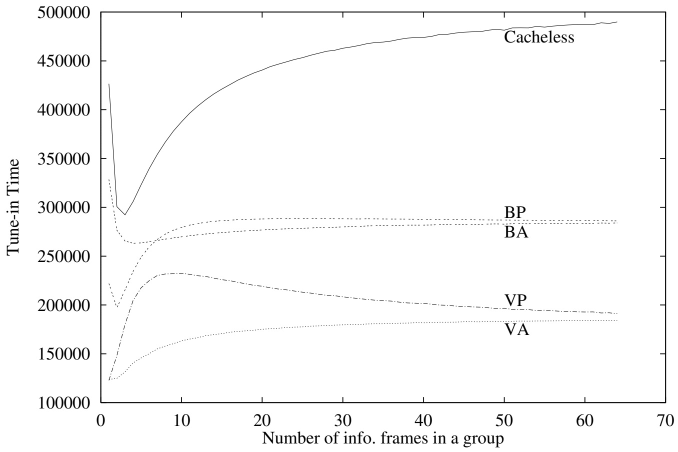

Figure 6: Tune-in time vs group size

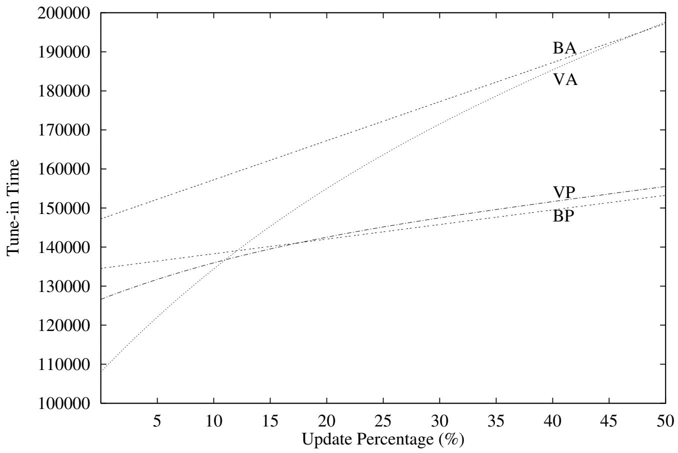

Figure 7: Tune-in time vs update percentage.

to observe its effect on caching policies. In this comparison, we set the locality of the modified frames in a frame group to 2 in order to allow the percentage of the updated frames to go up to 50%. Figure 7 shows the tune-in time performance. As the update percentage increases, the caching methods using passive refreshing strategy perform better than the methods using active strategy. This is because the passive strategies only load the signatures needed for query processing but not residing in the cache, while the active policies try to maintain all of the signatures in cache.

6 Limiting The Cache Size

In the previous sections, we assumed that the cache size is large enough for accommodating all of the cached signatures. However, some of the mobile computers may have rather small memory for caching signatures. Thus, replacement policies have to be considered for mobile computers with small cache memory.

6.1 Cache Replacement Policies

Two factors of the broadcast and filter services have to be considered when we design replacement policies: access frequency and update frequency. Update frequency is decided by the information servers at the fixed network. However, the mobile computers may collect it, along with the access frequency, by statistical approach.

Due to the sequential scanning nature of the signature schemes, all of the integrated signatures have to be compared with the integrated query signature in order to decide whether their associated simple signatures are needed for subsequent comparisons. Therefore, user access patterns do not have an impact on the cost of integrated signature comparison. As such, a simple but feasible cache replacement policy is to keep as many integrated signatures in cache as possible. When a signature is updated, its slot is filled with the next integrated signature not in cache. The policy is to let some infrequently updated integrated signatures eventually get into the cache.

Intuitively, the above policy makes full use of the cache available when the cache size is not large enough to hold all of the integrated signatures. When the cache size is large enough, the simple signatures should be considered for caching too, especially those corresponding to frequently matched integrated signatures.

To cache a mix of the integrated and simple signatures, we adopt the policy which replaces the most frequently updated and least frequently accessed signatures. Integrated signatures are more frequently accessed than most of the simple signatures. However, integrated signatures also have higher average update rate than simple signatures do. This is because when any information frame in the frame group is updated, the corresponding integrated signature has to be updated while only the simple signature corresponding to the updated information frame in the group has to be updated. Therefore, an integrated signature may be moved out of the cache while some of the simple signatures in its frame group may stay in the cache.

The approach we proposed for general signature caching combines passive refresh policy and the replacement policy described above. The refresh strategy is to decide when to execute the replacement policy, while the replacement policy is to decide which signatures should stay in the cache. Based on the passive refresh policy, a signature will be considered for caching only when it is received over the channel. For the replacement policy, a *replacement score* (RS) is maintained for each signature to decide which signatures are to be cached.

When a mobile computer starts the filtering process, it will synchronize to the broadcast channel. From then on, for each incoming signature, it checks whether the signature is updated, in cache, and needed for signature comparison. The following describes the actions taken under various situations:

- if the signature is not used for signature comparison and not updated, then the mobile computer goes into doze mode.
- if the signature is not needed for signature comparison, but it is updated:

– if the signature is in cache, then delete the signature from cache, update its replacement score, and go to doze mode.
– otherwise, directly update its replacement score, and go to doze mode.

- if the signature is needed for signature comparison and it's not in the cache, the mobile computer will receive the signature from the channel, compare it to query signature, update its replacement score, and execute the cache replacement policy.
- if the signature is needed for signature comparison and it's in the cache,
  – if the signature is updated, receive the signature from the channel, do comparison and update replacement score.
  – if the signature is not updated, do comparison on cached signature; update replacement score and go to doze mode.

The cache replacement algorithm maintains a replacement score for each signature in the broadcast cycle:

$$RS = \text{number of comparisons} - \omega \cdot \text{number of updates}.$$

where ω is the weight of update to comparisons and $0 \leq \omega \leq \infty$.

The replacement scores are updated when a signature is updated or is received over the channel for comparison. When a signature is received over the channel, it will be considered for caching. Its replacement score is compared with the lowest score of the cached signatures. If the score is higher than the lowest score, the signature is brought into the cache to replace the signature with the lowest score.

6.2 Performance Evaluation of the Cache Replacement Policy

In the following, we simulate the signature cache replacement policies. In the simulation for the policy of caching both simple and integrated signatures, we assume the weight for the number of updates, ω, to be 1. In our experiments, we fix the size of a frame group to 4. We assume the selectivity of queries (i.e., true match probability) to be 5%, update probability to be 1%, and false drop probability of the integrated signatures to be 1%. The localities for true drops and updates are set to 2. Initially, the cache is empty, but it is gradually filled in the subsequent cycles and they have increasing hit rates. In the simulation, we measure the number of hits from the initial query up to the 25th query. To observe the effect of cache size on the number of hits, we repeat the same experiment for various cache sizes.

Figure 8 shows the number of hits for different cache sizes. On the top of the figure, the curve labeled as 'reference' is the number of signatures needed for comparisons. In other words, it represents the number of signatures received from the communication channel if no cache is used. Other curves are labeled by the policy

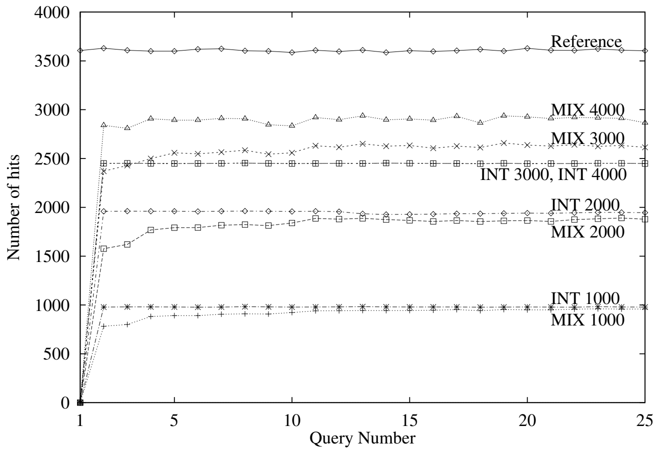

Figure 8: Hits vs query number.

and cache size used. For example, 'INT 1000' represents the cache hits for the pure integrated signature cache replacement policy with cache size of 1000 signatures, whereas 'MIX 2000' represents the cache hits for the mixed signature cache replacement policy with cache size of 2000 signatures. It can be observed that in every configuration, the initial query has no cache hit but the second query rapidly increases the hits. Then, the cache hits for the MIX experiments gradually increase for the subsequent queries, while the cache hits for INT experiments maintain at the same levels. In the figure, 'INT 3000' and 'INT 4000' are overlapped, because there are only 2500 integrated signatures in our experiment. Therefore, the INT method with cache size greater than 2500 signatures has the same cache hit results.

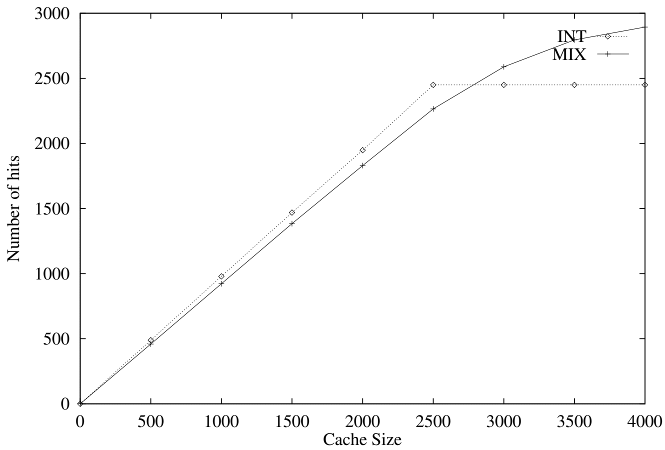

Figure 9: Hits vs cache size.

Figure 9 shows the average number of hits over the 25 queries for different cache sizes. From the figure, we observe that, for the INT method which caches only the integrated signatures, the number of cache hits is proportional to the cache size up to 2500, the total number of integrated signatures used in the simulation. From then on, increasing the cache size makes no difference, because all of the integrated signatures are already cached.

On the other hand, for the MIX method, the number of hits increases steadily and is proportional to the cache size. When the cache size is over 2500, however, the increase slows down but continues. The phenomenon may be explained as follows. At the beginning, as many integrated signatures as possible will be brought into the cache. Since integrated signatures are always used for comparison, the hit rate is high. As most of the integrated signatures are in cache, the number of simple signatures in cache increases. Whether a simple signature is used in the filtering process depends on the selectivity and the false drop probability of the integrated signatures. Therefore, the hit rate of the simple signatures is not as high as that of the integrated signatures. Thus, the increase of hits slows down when the cache size is greater than 2500. The same reason also explains why the MIX method has slightly lower hit rates than the INT method when the cache size is smaller than 2500 signatures.

7 Conclusion

Since most queries on broadcast information select only a small number of information frames, indexing is very effective in reducing the tune-in time. With a reasonable false drop probability and small signature overhead, the signature scheme is excellent for information filtering in information broadcast services. Compared to traditional indexing, the signature method is particularly suitable for mobile computers because it can perform realtime filtering with little processing and memory requirement.

This paper describes policies for caching signatures in mobile computers. We propose four caching policies for the two-level signature scheme. Unlike the performance consideration of traditional disk accesses, tune-in time, which corresponds to battery consumption in the operation, and access time are used as performance criteria in evaluating information filtering methods. The cost models for the tune-in time and access time of the caching policies have been developed and compared based on various factors.

With reasonable access time delay, the two-level signature scheme with various caching policies outperforms the same scheme without caching. The policies using version numbers are in general better than those using bit tags. As the percentage of updated frames increases, the policies using passive refreshing strategy is better than that using active strategy.

We also discuss the cache replacement policies for mobile computers with small caches. Two replacement policies are proposed, one caches integrated signatures only and the other caches both simple and integrated signatures. We simulate the two policies to compare their cache hits. The result shows that, for small cache, the policy caching only integrated signatures is better, while the policy caching both kinds of the signatures has a comparable performance. On the other hand, when the cache size is larger than that of the integrated signatures, the policy caching both kinds of signatures is better.

For future research, we will investigate caching policies and optimization issues involved with data and channel allocation for other kinds of mobile applications, e.g., those allow mobile users to transmitting queries to the information servers.

References

[1] S. Acharya, R. Alonso, M. Franklin & S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments," Department of Computer Science, Brown University, Technical Report No. CS-94-43, December 1994.

[2] D. Barbara & T. Imielinski, "Sleepers and Workaholics: Cashing Strategies in Mobile Environments," *Proceedings of the International Conference on SIGMOD*, May 1994, 1–12.

[3] W.W. Chang & H.J. Schek, "A signature access method for the Starburst database system," *Proceedings of the Fifteenth International Conference on Very Large Data Bases*, Amsterdam, The Netherlands, Aug. 1989, 145–153.

[4] C. Faloutsos & S. Christodoulakis, "Signature files: an access method for documents and its analytical performance evaluation," *ACM Transactions on Office Information Systems*, Vol. 2, No. 4, Oct. 1984, 267–288.

[5] S. Ganguly & R. Alonso, "Query Optimization in Mobile Environments," Department of Computer Science, Rutgers University, LCSR-TR-219, 1993.

[6] D.J. Goodman & T. Imielinski, "Future directions for wireless data," *Digest of Papers. Spring COMPCON 94*, March 1994, 464–466.

[7] T. Imielinski & B. R. Badrinath, "Mobile Wireless Computing: Solutions and Challenges in Data Management," Department of Computer Science, Rutgers University, DCS-TR-296, 1993.

[8] T. Imielinski, S. Viswanathan & B. R. Badrinath, "Power Efficiency Filtering of Data on Air," *Proceedings of the International Conference on Extending Database Technology*, 1994, 245–258.

[9] T. Imielinski, S. Viswanathan & B. R. Badrinath, "Energy Efficiency Indexing on Air," *Proceedings of the International Conference on SIGMOD*, May 1994, 25–36.

[10] W.-C. Lee & D.L. Lee, "Information Filtering in Wireless and Mobile Environments," *Proceedings of the 15th Annual IEEE International Phoenix Conference on Computers and Communications, Phoenix, Arizona*, March 27-29, 1996, to appear.

[11] R. Sacks-Davis & K. Ramamohanarao, "A two level superimposed coding scheme for partial match retrieval," *Information Systems*, Vol. 8, No. 4, 1983, 273–280.