

# Approaches for Broadcasting Temporal Data in Mobile Computing Systems

Kam-Yiu Lam, Edward Chan and Joe Chun-Hung Yuen  
Department of Computer Science  
City University of Hong Kong  
83 Tat Chee Avenue, Kowloon  
Hong Kong  
Fax: 852-2788-8614  
Email: {cskylam|csedchan|csjyuen}@cityu.edu.hk

## *Abstract*

*Rapid advances in mobile communication technology have spawned many new mobile applications. A key element in many of these systems is the need to distribute real-time information from a database server to mobile clients. While data broadcast has been shown to be an efficient data dissemination technique, many issues such as selection of broadcast data and caching strategies at mobile clients are still active research areas. In this paper, we consider an important characteristic of many mobile computing systems which has often been ignored in the design of broadcast algorithms: the fact that many data items are associated with temporal constraints on their validity. We introduce the notion of Absolute Validity Interval (AVI) to capture the temporal constraints of the data items, formulate a temporal data model and examine both static and dynamic approaches to select data items based on their access frequencies as well as their AVI. The reason for considering the AVI of the data items in broadcast selection is to increase the client cache hit probability so that the access delay for a data item will be much reduced. Based on the results from extensive simulation experiments, it is concluded the AVI-based approaches, by improving cache hit probability, can significantly improve the mean response time and reduce the number of deadline missing requests.*

Keywords: mobile computing, data broadcast, real-time data, cache management

## **1 Introduction**

Recent advances in wireless communication technology have greatly increased the feasibility of mobile information services. At the same time, mobile computing units such as notebook computers and palmtop machines are making impressive inroads into the mass consumer market. A number of novel mobile information services such as mobile shopping aids in large shopping malls and financial information distribution to users via mobile phones have already been implemented. Innovative applications, such as real-time traffic information and navigation systems, and real-time stock information systems, are emerging as data-hungry users require instant access to information

using their palmtops and PDAs no matter where they are located.

Mobile computing systems are characterized by a number of constraints such as high transmission error rates and low bandwidth, as well as limited power supply for many client systems. As a result, the design of an efficient and cost effective mobile computing system requires techniques that are quite different from that in other distributed systems based on wired networks (Imielinski and Badrinath, 1994; Pitoura and Bhargava, 1993; Pitoura and Samaras, 1998). Recent research has focused on one of the most important issues: supporting efficient data retrieval and dissemination to the requests from mobile clients (Imielinski and Badrinath, 1994; Acharya, 1997; Acharya et al, 1997). Here, the performance objective is to reduce the mean access delay for data items and minimize the response times of the requests from the mobile clients. Generally speaking, the proposed mechanisms for data dissemination to the mobile clients fall into one of two major approaches: the *on-demand* approach and the *broadcast* approach (also called the *pull* and *push* approaches respectively) (Acharya, 1997; Acharya et al, 1997; Franklin and Zdonik, 1998).

In the on-demand approach, the data items required by the requests from mobile clients will be sent from the information server on request. This approach is simple but does not scale well with the number of mobile client requests. Given the limited bandwidth currently associated with most mobile computing systems, the waiting time for a data item can be very long if there are a large number of requests waiting for different data items (Acharya, 1997; Xuan et al, 1997). On the other hand, in the broadcast approach, the information server periodically and continuously broadcasts data items to the mobile clients. A mobile client waiting for a data item can simply get the data item from the “air” while it is being broadcast. Using this approach, the cost for data dissemination is independent of the number of requests since a broadcast can satisfy multiple requests for the same data item, resulting in much more efficient utilization of the bandwidth and much better scalability. As a result of these benefits, many of the previous work on data dissemination in mobile systems are based on the broadcast approach (Acharya, 1997; Datta et al, 1997; Hameed and Vaidya, 1997; Jiang Vaidya, 1998; Lee, Hu and Lee, 1997; Leong and Si, 1995; Shivakumar and Venkatasubramanian, 1996).

A related issue in data broadcast is the management of cached data (Acharya et al, 1997). Caching of data items can greatly reduce the access delay for a data item (Chan et al, 1998; Leong et al, 1996). Clearly, if the needed data items can be found in the client cache, the clients’ requests may be served immediately. A variety of strategies for cache management and invalidation in mobile computing systems have been proposed in the literature (Acharya et al, 1997; Hu and Lee, 1997; Jing et al, 1995; Wu et al, 1996). However, a drawback of these approaches is that they typically have high overheads as invalidation messages have to be sent to the clients whenever a data item at the server is

updated.

In this paper, we focus on the data dissemination problem using the broadcast approach. Perhaps the most important design issue in this approach is the method used for selecting data items from the database for broadcast. Although a number of methods which aims to minimize the average delay for data items (Acharya, 1997; Datta et al, 1997; Jiang and Vaidya, 1998; Lee, Hu and Lee, 1997; Leong and Si, 1997) have been proposed, most of them are based on the popularity, or the access frequencies, of the data items. However, in many mobile computing systems, a request from a mobile client is associated with a completion deadline, which can be either explicit or implicit. Even though the deadlines may only be “soft” deadlines, missing them can render the response to the mobile clients useless (Kayan and Ulusoy, 1999; Uulsoy, 1998). In addition to timing constraints on the retrieval of data items, we note that many data items in a mobile computing system are status information, for instance the last traded price of a stock or the current traffic situation of a road. Their values are highly dynamic as they are used to represent the current status of the objects in the external environment. The validity of the current values of the data items depends on the temporal constraints and they may become out-dated and useless with the passage of time.

Although a lot of work has been done on data dissemination for mobile computing systems, the temporal properties of data items have been largely ignored in the research literature. In this paper, we introduce a new data model to characterize the temporal properties of the data items. Based on this temporal data model, we propose new broadcast approaches, which we call AVI-based approaches, in which the temporal properties of the data items will be considered in selecting the data items for broadcast. We will show that using these approaches the client cache hit probability can be improved significantly so that the access delay for the data items can be much reduced.

The organization of the remaining parts of the paper is as follows. Section 2 reviews previous work on the broadcast approach for mobile computing systems. Section 3 introduces the temporal data model. Two new data broadcast strategies, which consider temporal properties of data explicitly, will be presented in Section 4. Evaluation of the broadcast strategies by means of simulation is the subject of Section 5, and the paper concludes with a brief summary in Section 6.

## **2 Related Work**

Research on data dissemination in mobile computing systems has received a lot of attention in recent years. As we have seen in the previous section, both broadcast and on-demand approaches have been studied fairly extensively in the literature. Generally speaking, the results have indicated that the broadcast approach can better utilize the limited bandwidth of a mobile computing system and

it is particularly suitable for dissemination of large volume of data to a large number of mobile clients at the same time (Acharya, 1997; Xuan et al, 1997).

Research in data broadcast mechanisms can be grouped into several areas. Some researchers have focused on the design and evaluation of efficient access methods. The key focus of this line of research is to optimize the broadcast data structure (ranging from indexing to signature and hashing) so that the request tune-in time as well as access delay are minimized (Hu, Lee and Lee, 1998; Imielinski et al, 1994; Imielinski et al, 1997; Shivakumar and Venkatasubramanian, 1996)

The design of data broadcasting strategies, such as the length of each broadcast cycle and the data selection mechanisms, is another major research area. (Leong and Si, 1997; Hameed and Vaidya, 1997; Stathatoas et al, 1997; Su and Tassiulas, 1997). Basically, most of the proposals use access frequencies of the data items to determine what should be selected for broadcast. A variety of selection algorithms have been proposed, which differ primarily on the assumptions made. For instance, most of the proposed algorithms assume that the access probabilities of data items are available. A notable exception is (Stathatoas et al, 1997), which uses a periodic probing mechanism to determine whether a particular data item is in demand or not.

Some researchers have proposed hybrid approaches in which periodic broadcast of data is combined with support for explicit client requests (Xuan et al, 1997; Datta et al, 1997). However, the performance of the hybrid approach does depend on careful allocation of bandwidth for the two types of data dissemination (Xuan et al, 1997). In (Fernandez and Ramamritham, 1999), an adaptive approach is suggested to allocate the bandwidth for data broadcast and the on-demand approach.

Another major difference between the proposed algorithms is the role of caching. In (Leong, Si and Chan, 1996; Leong and Si, 1997), the authors reason that mobile computers cannot afford to have large cache and introduce the concept of using the broadcast data as air-storage, a concept rather similar to the broadcast disk proposed in (Acharya, 1997). The data selection algorithm can then be recast as a cache replacement algorithm, and the key performance metric of these studies is now the hit-ratio instead of the more commonly used access time or tune-in time. An Exponential Weighted Moving Average (EWMA) approach is proposed to select data items. Researchers who do not consider the effect of cache view the data selection issue purely as a scheduling problem, and have formulated algorithms which can generate a broadcast schedule which will minimize the access time, provided that the access probabilities of the data items are known (Vaidya and Hameed, 1996).

Only a few papers in the literature have considered the real-time constraints of the client requests. A recent study (Xuan et al, 1997) presents a data selection and scheduling algorithm based on

the deadlines of the requests. Using an Earliest Deadline First (EDF) approach, each data item in the information server is associated with a deadline based on the timing constraints imposed by all the requests for that particular data item, and the data items with the earliest deadlines will be broadcast first. This strategy is shown to be more effective than the on-demand approach if the arrival rate of the requests is high.

In this paper we will extend the work of others by modeling the temporal properties of data items in greater detail. As we will see, associating client requests with deadlines does not capture all the temporal properties of the data. Our model is particularly useful in systems with client cache, since cached data may expire. Using our improved temporal data model, we will demonstrate how cached data can be refreshed at a relatively low cost using new broadcast strategies, thus leading to higher cache hit rates and lower access delay. We believe that our model will not only facilitate a more realistic and complete analysis of many practical mobile computing systems but also improve overall system performance as well.

### **3 Temporal Data Model for Mobile Computing Systems**

In formulating our data model for mobile computing systems, we assume that the data items in the database are used to record the instantaneous values of the objects in the external environment. Some examples are the last traded stock prices, news updates, as well as traffic and weather conditions. Since they track dynamic external events, this information may change quite rapidly. To maintain strict consistency of the data items with the actual status values of their corresponding objects in the external environment, it may be necessary to refresh the database records frequently.

The price of maintaining tight consistency between the data items in the database and the actual status of the objects in the external environment is that tracking every change in the status of the external objects will create a large number of update transactions, which may not be practical for many mobile computing systems. Actually, for many data items, such as those for weather condition, news updates and traffic situations, it is possible for updates to be performed at much longer intervals. There are two reasons for this. Firstly, for many mobile computing systems, the value of a data item that models an object in the external environment cannot, in general, be updated continuously to perfectly track the dynamics of the real-world object as the update process itself requires time to complete. Thus, there already exists a discrepancy between the value of the data item and the “real-time” status of its corresponding object. An example might be the traffic status of a particular road. In this case, by the time this information is reported to the server, the actual traffic conditions of the road will already have changed. Secondly, it is often unnecessary for data values to be perfectly up-to-date or precise to be useful. In particular, values of a data item that are slightly different are often

interchangeable and the mobile clients will consider them to be the same. For example, the traffic condition at one minute past noon can be considered to be the same as the traffic condition at noon to many drivers.

The dynamic properties of the data items also make the management of cached data items a difficult problem (Acharya et al, 1997). Whenever a new version of a data item has been created in the database, all the cached versions of the data item have to be invalidated. Traditionally, invalidation is achieved by sending invalidation messages to the clients which have cached the data item. However, due to the delay in invalidation as a result of the low bandwidth in mobile environments, it is difficult to maintain a tight consistency between the data items at the client cache and the data items in database. Thus the key problem is that even when the required data item is located in the client cache, it is necessary to determine whether this is the most updated version without incurring heavy overhead.

To resolve these issues, we introduce a new model to characterize the temporal properties of the data items in mobile computing systems. In our model, it is assumed that the data items are classified into different *data groups* based on one of their temporal properties, namely the rate at which their values change. Within each data group, the data items are assigned a time frame, called *Absolute Validity Interval (AVI)*, to characterize this property. The idea of absolute validity interval originates in the studies in real-time database systems in which the values of the data items are highly dynamic (Ramamritham, 1993). Formally, each data item can be defined as a 3-tuple:

$$D_i = (V_i, AVI_i, TS_i)$$

where  $V_i$  is the value for data item  $D_i$ ,  $AVI_i$  is the absolute validity interval of  $D_i$  and  $TS_i$  is the last update time-stamp of  $D_i$ . Each transaction update, which captures the “real-time” status of an external object, is associated with a time-stamp, called *update time-stamp (TS)* which indicates at which time the update transaction is created. Whenever an update transaction is completed, the update time-stamp will also be recorded in the database along with the updated data value. A data item,  $D_i$ , is *invalid* if  $AVI_i + TS_i < \text{current time}$ . In this case, a new value of the data item is required.

The choice of AVI values for the data items depends on the nature of the data items. It is similar to the concept of *data similarity* (Kuo and Mok, 1992; Kuo and Mok, 1993). It is assumed that the application semantics allows the system designer to derive a similarity bound for each data item such that two write operations on the data item will be *similar* if the values come from a time interval not greater than the similarity bound. In the AVI approach, the AVIs of data items can be defined as the similarity bound of the data items. Their values can be based on the dynamic property of the data

items, how the mobile clients use the data items and the importance of the data items towards the mobile clients. For example, news updates are generated every 30 minutes. Thus, the AVI of news updates can be defined to be 30 minutes. For information whose update is asynchronous (examples include stock prices), their AVI can be defined based on the user requirements. For example, free delayed stock quotations might have a large AVI (in the order of minutes) while real-time stock information used by professional traders or paid quotation services might have a smaller AVI (probably on the order of seconds). Another type of data with a larger AVI is traffic information where changes are less drastic than stock prices.

## 4 AVI-based Broadcast Approaches

In this section, we will examine our proposed AVI-based broadcast strategies in some detail. We note that there are two main components in the broadcast strategies:

- (1) Selection of data items: determination of which data items should be selected for broadcast in each broadcast cycle; and
- (2) Broadcast skipping: even if a data item has been selected for broadcast, its broadcast may be skipped if it has been broadcast in the recent broadcast cycles. The reason is that it may still be valid in the client caches. The purpose of broadcast skipping is to save broadcast bandwidth so that more data items can be broadcast in the same period of time.

### 4.1 Selection of Data items in the AVI-based Approaches

As we have indicated in the previous section, the primary issue in the design of broadcast algorithms is to determine which client requests should be served first under the limited bandwidth. As there are a large number of client requests for different data items, given the limited capacity of the broadcast channel the server may not be able to satisfy all the requests before their deadlines. Even assuming, as in many previous work, the server knows the previous data access patterns of the requests and which data items are needed by the current requests (presumably by sampling or by asking the mobile clients to periodically generate reports to the server), it is still not easy to determine an optimal schedule to broadcast the data items so that all the requests can be served before their deadlines or the number of deadline missing requests can be minimized. The selection problem is a knapsack problem and is NP-hard and that the optimal schedule depends on the distribution of the required data items of the requests and their deadlines (Xuan et al, 1997).

In our AVI-based broadcast strategies, both the access frequencies *and* AVI of the data items are considered in selecting the data items. Using the access frequencies of the data items is a powerful and effective techniques used in a number of previous studies (e.g. Leong and Si, 1997). However, we believe that the performance of this technique can be further improved by considering the AVI of the data items, so that invalid items can be removed from the cache and possibly replaced by updated versions. In the following sections, we first describe pure access frequency based approaches, and then proceed to propose two versions of AVI-based broadcast strategies.

#### 4.1.1 Access Frequency-based Broadcast (AF)

In this approach for data broadcast, each data item is assigned a score based on its access frequency. The needed statistics can be obtained by sampling or by asking the mobile clients to generate access reports periodically to the server. In the calculation of access scores, we use the method suggested in (Leong and Si, 1997), called Exponential Weighted Moving Average (EWMA). In this method, a weighed moving average is used to calculate the access scores of the data items. The highest weight is given to the data items which are requested by the current requests; the weights assigned to the previously requested data items tail off as they age. The most current requests get a weight of 1, while those in the previous cycle receives a weight of  $\alpha$ , which is a tunable parameter and its value has to be smaller than one. The data items, which have been requested in the next previous cycle, are given a weight of  $\alpha^2$  and so on. The final access score of a data item is obtained by summing up all its scores in each broadcast cycle and then normalized through dividing the sum of all the weights used:

$$f_{x,n} = (f_{x,n} + \alpha f_{x,n-1} + \alpha^2 f_{x,n-2} + \dots + \alpha^{n-1} f_{x,1}) / S$$

where S is the sum of the weights  $(1 + \alpha + \alpha^2 + \dots + \alpha^{n-1})$

#### 4.1.2 Static AVI-based Broadcast (SAVI)

Basically, there are two ways to include the AVI of data items into the AVI-based broadcast approaches. The first one is a static approach in which a fixed bandwidth, defined in terms of number of data items, is reserved for the data items in each data group based on the AVI value of the data group. Note that here we assume that the data items have similar sizes. If their sizes are different, the bandwidth can be defined in time unit. The selection of data items from a data group is then based on their access scores. The second approach is a dynamic approach in which a weighting factor, based on the AVI of the data items, is included in the calculation of the access score for each data item.

In the Static AVI approach, the selection of data items for broadcast is divided into two steps. The broadcast cycle is first divided into  $n$  partitions,  $C_1$  to  $C_n$ , where  $n$  is the number of data groups in the database.  $B_i$  is the size of partition  $C_i$ .  $B_i$  is defined to be proportional to  $1/AVI_i$ .

$$B_i = DS \times \left( \frac{1}{av_i} \right) \times \frac{S_i}{DB} \sum_{j=1}^k \frac{1}{av_j}$$

where  $k$  is the number of data groups in the system,  $DS$  is the size of a broadcast cycle,  $DB$  is the size of the database and  $S_i$  is the size of the data group  $i$ . Note all of these requests are defined in terms of number of data items.

After the bandwidth to be assigned to a data group has been determined, the selection of the data items from a data group will be based on their access scores. The data item with a higher score will be selected first until the bandwidth (i.e.  $B$ ) assigned to the data group has been used up.

In SAVI, the data items with shorter AVI are favored in the data selection process. The basic rationale for doing this is that data items with short AVI remains valid in the client cache for only a short period of time. In order to maintain validity of data items and hence improve client cache hit rate, these data items are given higher priorities in data selection.

#### 4.1.3 Dynamic AVI-based Broadcast (DAVI)

In SAVI, if the AVI of the data groups are very different, the bandwidth (in terms of the number of data items to be selected from that group) assigned to the data groups will also be very different. It is possible that the bandwidth allocated to a data group with a very large AVI will be very small especially when the length of the broadcast cycle is short. Thus, only a few data items can be selected from this group for broadcast in each broadcast cycle even though the access scores of the data items in that group are high compared with the data items in the other groups which have much smaller AVI values.

In the Dynamic AVI approach (DAVI), the above problem is solved as the bandwidth allocated to each data group is not fixed. In the selection of data items for broadcast, both the AVI of different data groups and the access frequencies of the data items will be considered at the *same time*. They are used to calculate a final access score for a data item,  $D_i$ :

$$\text{Score}(D) = f(D) \times \left(1 + \left(\frac{1}{\sum_{j=1}^k \frac{1}{\text{AVI}_j}}\right)^w\right) \times \frac{S_i}{DB}$$

where  $f(D_i)$  is a function used to calculate the access score of the data item  $D_i$  based on its access frequency from the mobile clients and  $w$  is a tunable parameter to define the weighting (importance) of the AVI of the data items as compared with the access frequency function  $f(D_i)$ .

## 4.2 Broadcast Skipping

If a data item has been broadcast in the last broadcast cycle, it may not be necessary to broadcast it again in the current broadcast cycle if it is still valid at the end of the current cycle. Client request can simply be satisfied using the version found in the client cache. We can just skip the data item and use the saved bandwidth to send other data. We call this technique *broadcast skipping*.

However, if a data item has not been cached at a mobile client, the use of broadcast skipping may cause the requests from the mobile client to wait for a long period of time as their data items may be skipped in the current broadcast cycle. To prevent this, we stipulate that a data item will be skipped for broadcast only if there is no request waiting for it:

```

For each data group
  Number of selected data item from the data group = 0
  While number of selected data item from the data group <
    no of data item to be selected from the data group
  Do
    Get the data item with highest score from the unselected data item set
    If ((the completion time of this broadcast cycle –
      last broadcast time of the data item) >
      AVI of the data item) or
      (the number of requesting waiting for the data item > 0)*
      Select the data item into the broadcast data item set
      Increment the number of selected data item by one
    Endif
    Remove the data item from the unselected data item set
  End While
End For

```

The determination of data skipping for a data item depends on the last broadcast time of the

data item and whether it is requesting by a mobile client. Thus, a data item will be skipped if its last broadcast time is not smaller than the current time by its AVI even though it is broadcast in several cycles before. This information and the information of which data item is being requesting can be obtained by sampling or by asking the mobile clients to generate access reports to the server periodically. Note that the reports are already required in the data broadcast approaches, which are based on access frequencies of data items (Chan, Si and Leong, 1998; Leong and Si, 1997; Xuan et al, 1997; Datta et al, 1997). Thus, the introduction of broadcast skipping should not incur much additional overhead.

### **4.3 Discussion and Advantages**

An important advantage of using AVI to define the validity of a data item is that it can make the management of the cached data at the client caches much easier. As each data item is associated with an AVI, invalidation message is no longer needed to invalidate the data items in the client caches since the validity of the data items is explicitly defined by their AVI and their last update time-stamps. When a request has found that its required data item is at the client cache, it can determine the validity of the data item immediately by checking the data item's time-stamp and AVI.

The problem of the AVI approaches is that a data item may become invalid while it is in the client cache. Without a suitable cache refreshment policy, the probability of cache hit (finding valid data items in the cache) will decrease over time, leading to an increase in access delay. However, if the data items are not selected for broadcast, it is not possible to refresh their validity in the client cache under the broadcast approach. Of course, the validity of the data items at the client caches can be maintained by the on demand approach or prefetching (Acharya et al, 1997). However, this will result in an inefficient use of the bandwidth and is undesirable in mobile computing systems. It is clear that in this case data broadcast serves not only to satisfy user request directly but also to improve cache hit rates as maintaining valid data items in the client cache can greatly reduce the data access delay. Thus, if the broadcast algorithm can consider the temporal constraints of the data items (AVI), it is expected that the validity of the cached data items can be better maintained and the delay for accessing a data item can be much reduced. Another important property of the AVI approach is that it can save broadcast bandwidth by broadcast skipping. This is especially important if some of the data items are hot data items.

## **5 Performance Study**

In this section, we evaluate the performance of both SAVI and DAVI, using simulation as our main tool. They are then compared with approaches based purely on Access Frequencies (AF). We use EWMA (Leong and Si, 1997) in both the AVI approaches and AF approach to calculate the

access scores of the data items to facilitate comparison. The simulation model as well as the details of the simulation process will be described first in the next section.

## 5.1 Simulation Model

Figure 1 depicts the model of the mobile computing system used in our simulation experiments. It consists of an information server and a number of mobile clients. The mobile clients are connected through a wireless network to the information server which maintains a database to record different real-time information of the external environment. It is assumed that the data items in the database are classified into  $N$  data item groups based on their AVI. The data items belonging to the same group will have the same AVI value.

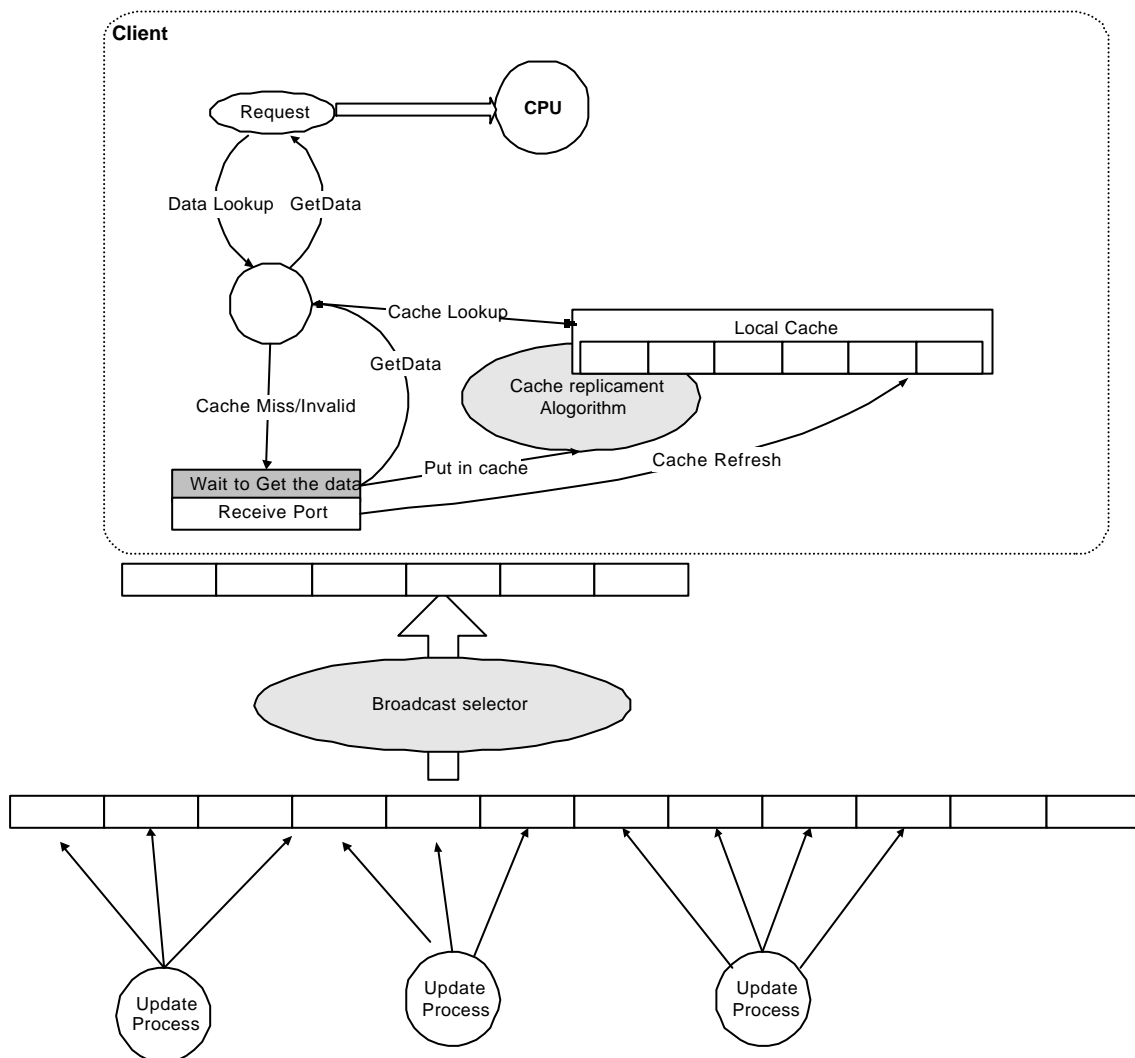


Figure 1: Model of the Mobile Computing System

The information server is connected directly to a number of update processes which generate updates to refresh the validity of the data items in the database. It is assumed that each update process

is responsible for the update of the data items in a data group. An update randomly selects a data item in the group to update. The creation time of the update will also be recorded in the data item. The generation of the updates is periodic. The period of an update process is defined to be the AVI of the data group (which the update process is going to update) divided by the number of data items in the data group.

The information server broadcasts data items from the database one by one until the end of a broadcast cycle, at which point it starts another broadcast cycle immediately. The selection of data items to broadcast is performed by the broadcast selector in which a broadcast selection approach is implemented.

In addition to the process for the information server, a number of processes are created and each such process is used to model a mobile client which generate requests for different data items. It is assumed that each request is defined with a drop period. If it cannot be completed at the time equals to its generation time plus the drop period, it will be dropped as the requesting information will be assumed to be useless. After the completion of a request, the mobile client will generate the next request after a think time. It is assumed that each request requires to access one data item.

A client cache is maintained at each mobile client. In processing a request, the cache of the mobile client will be searched first. The request is satisfied immediately if the required data item is found in the client cache, otherwise the request will be blocked until the required data item is being broadcast. After getting the required data item ‘from the air’ eventually, the request will be processed by the CPU. The data item will also be put in the cache where it will be available for future requests. If the cache is full, a cache replacement algorithm will be invoked to free some space for the incoming data item. In our model, we use a modified version of Least Recently Used (LRU) for cache replacement. In this technique which we call Invalid-LRU (I-LRU), the invalid data items will be selected for replacement first. If all the data items are valid, replacement is done using the original LRU principle. Note that if the data item, which is being broadcast, also exists in the cache, the copy in the cache will be refreshed by the new version. The reason of choosing a policy based on LRU is that LRU had been found to give a good performance in cache replacement in traditional cache system and it also had been shown to perform well for data broadcast cache replacement (Acharya, 1997).

The required data items of the requests are distributed in the database and are generated randomly. It is assumed that the requests from mobile clients have locality in the references to data items. The locality of a mobile client is defined by two variables, *DataRange* and *AccessProbability*. *DataRange* defines the set of hot data items of a mobile client. *AccessProbability* defines the access

probability of the mobile client on the hot data items. Each client has a different set of hot data items and the hot data items of different clients are evenly distributed among different data groups in the database.

## 5.2 Model Parameters

The following lists the model parameters and their baseline values:

<b>Parameters</b>	<b>Baseline values</b>
Broadcast cycle length	200 data items
Database size	4000 data items
Number of data item groups	4
AVI of the group 1 data items	350 sec
AVI of the group 2 data items	750 sec
AVI of the group 3 data items	1400 sec
AVI of the group 4 data items	2800 sec
Number of mobile clients	400
Broadcast bandwidth	18kbps
Size of a data item	1.8kbit
CPU service time for a request	1.0 – 5.0 ms (uniformly distributed)
Think time	0 second
Cache size	25 data items
Weighting factor, $\alpha$ , for calculating access frequency	0.5
w, weighting factor for DAVI	1
Proportion of hot data items	0.001
Access probability	0.8
Drop period	7.5 sec

Some of the parameter settings are defined based on the settings adopted in the previous research results, e.g., cache hit probability and cache size (Leong and Si, 1997). Some parameter values are determined according to the common characteristics of many mobile computing applications. For example, the bandwidth of the wireless communication link is chosen to be 18kbps which is the typical bandwidth for a radio cellular network. Due to the low bandwidth of mobile computing systems, the size of the data items cannot be large. Otherwise, it will take a long time to broadcast a data item and the performance of the system will be very poor. The size of the data items is chosen to be 1.8kbit. This is the typical size of simple text data such as the stock information and traffic news. Thus, the time required to broadcast a data item is 100ms. Some parameters are set to reflect the purposes of the performance studies of the paper. For example, in order to evaluate the performance of the proposed approach, different drop periods are tested.

It is assumed that in the baseline setting each data group has the same number of data items

and each group is defined with an *AVI*. In the determination of the *AVI* for the data groups, some of the data groups are assigned with small *AVI* values, such as 350 sec, as they are used to represent the highly dynamic data items, i.e., stock data, and larger values, such as 2800 sec, are assigned to the data groups with much less dynamic data items, e.g., the news updates. In the experiments, we also study the impact of different *AVI* values of the data groups and the number of data groups on the performance of the data broadcast approaches. The think time is set to zero in order to create a heavy workload so that at each time there will be about four hundred requests in the system.

### 5.3 Performance Measures

We use the following four performance measures in our experiments:

$R_R$  : mean response time of a request  
 $D_R$  : drop rate  
 $C_H$  : cache hit rate  
 $C_I$  : cache invalid rate

$R_R$  is defined as the response time of those requests which can be completed before their drop times.  $D_R$  is defined as the number of dropped requests divided by the total number of requests generated. It measures the capability of the system in meeting the deadlines of the requests. In addition to  $R_R$  and  $D_R$ , we also measure the cache hit rate and cache invalid rate.  $C_H$  is defined as the number of requests whose required data items are located in the client caches at the arrival time of the requests and are valid, divided by the total number of requests generated.  $C_I$  is defined as the number of requests, where required data items are in the client caches but are invalid, divided by the sum of the number of cache hit and the number cache invalid.

### 5.4 Results and Discussions

Our simulation program is implemented using the CSIM 18 simulation language (Mesquite, 1997). The length of simulation for each run is 160,000 seconds. The simulation length is determined after a series of trial runs until the results obtained are stable. For each simulation run, we use a batch mean method to collect the statistics. The statistics collected during the broadcast of the first 100,000 seconds are discarded in order to remove the initial bias. After the removal of the initial bias, 10 batch of results are collected and the length of each batch is equal to 5000 seconds. The final results are obtained from the mean values of the 10 batch results. For the drop rate and the mean response time of the requests, we have found that the confidence levels for 90% is within  $\pm 0.05\%$  of the mean values.

#### 5.4.1 Impact of Broadcast Cycle Length

Figure 2 to Figure 5 show the results for the three broadcast approaches at different broadcast cycle lengths. In Figure 2 and Figure 3, it can be observed that the *AVI* approaches consistently give a

better performance than AF in terms of mean response time,  $R_R$ , and drop rate,  $D_R$ . Amongst the three approaches, SAVI gives the best performance. When we compare the performance of SAVI with AF, we can observe that  $R_R$  for SAVI is about 15% smaller than that for AF.  $D_R$  for SAVI is also consistently about 4% smaller than that for AF. Although the performance of the DAVI is not as good as SAVI, its  $R_R$  is also about 10% smaller than that for AF when the broadcast length is smaller than 75 data items.

The better performance of the AVI approaches, especially SAVI, is due to higher cache hit rate,  $C_H$ , and smaller cache invalid rate,  $C_I$ , as can be observed in Figure 4 and Figure 5. The lower  $C_H$  for AF is due to higher  $C_I$  as can be observed in Figure 5. In both DAVI and SAVI, due to emphasis placed on the AVI of the data items in the data selection process, the data items with smaller AVI values are broadcast more frequently than the data items with larger AVI values. Thus, the validity of the data items in the client caches can be better maintained. As shown in Figure 5,  $C_I$  for SAVI is almost zero. Although the cache invalid rate for AF is also very small, however, a few percent drop in the cache invalid rate can greatly improve the system performance as the access delay for cache data items is much shorter than the access delay for data broadcast. These results confirm our belief that by considering the AVI of the data items in data selection the overall system performance can be significantly improved.

When we compare the performance of SAVI with DAVI as in Figures 2 to 5, it can be seen that the performance of SAVI is better than DAVI especially when the broadcast cycle length is long. Although the DAVI approach has greater freedom in choosing data items for broadcast, it is difficult to define what should be the contribution of the AVI of the data items in calculating the score for a data item as compared with the access frequencies of the data items. As observed in Figure 4, when the broadcast cycle is very small (e.g., the broadcast cycle length is smaller than 25), the performance of DAVI is marginally better than SAVI. It is because if the broadcast cycle length is very small, the bandwidth (defined in terms of number of data items) assigned to a data group will be very small especially for the data group with a large AVI. Thus, a “hot” data item may not be selected for broadcast even though its access score is very high. Thus, the performance of SAVI is affected. This effect decreases when the broadcast cycle length lengthens.

#### 5.4.2 Impact of AVI Values and Number of Data Groups

In this set of experiments, we increase the AVI of the four data groups to 700sec, 1400sec, 2800sec and 5600sec respectively. The impact of the broadcast cycle length on the performance of the three approaches is presented in Figures 6 to 8. It can be seen from Figures 6 and 7 that the performance of the three broadcast approaches is similar. It is because when the AVI of the data items are larger, the cached data items are usually valid. Thus,  $C_I$  for the three broadcast approaches are

about the same and approach zero as shown in Figure 8.  $C_i$  for AF is slightly greater than that for DAVI and SAVI.

The impact of the number of data groups on the performance of the broadcast approaches is shown in Figures 9 to 11. (When the number of data group is varied, the AVI of the new data groups are defined based on largest and smallest AVI values used in the baseline setting. For example, when two data groups are used, their AVI are 350 sec and 2800 sec.). Consistent with the results in section 5.4.1, the performance of the AVI approaches is better than AF and SAVI gives the best performance as shown in Figure 9 and Figure 10. Once again, the better performance of SAVI is due to low cache invalid rate as shown in Figure 11.  $C_i$  for SAVI remains zero but  $C_i$  for AF is more than 2.5% when the number of data group is two. The poor performance of AF when two data groups are used is due to large number of data items with a small AVI value, which results in serious cache invalid problems.

#### 5.4.3 Impact of Drop Period

The drop period defines the maximum waiting time of a request for its data item. A smaller drop period means that the requests have tighter deadlines. Figure 12 and Figure 13 depict the values of  $R_R$  and  $D_R$ , respectively, for the three approaches at different drop periods when the broadcast length is 200 data items. The performance of the three approaches at different drop periods is consistent with the results in Figure 2 and Figure 3. The performance of SAVI is better than DAVI which in turn is better than AF. As depicted in Figure 12,  $R_R$  increases with the drop period. A longer drop period allows the requests to wait longer for their data items. Thus,  $R_R$  increases with larger drop periods. Due to the looser deadline constraints, the value of  $D_R$  decreases slightly with an increase in drop period as shown in Figure 13. The better performance of the AVI approaches over AF is again due to the higher cache hit rate and smaller cache invalid rate as can be seen in Figure 14 and Figure 15.

Figures 16 and 17 shows the impact of drop period on the performance of the three approaches when the broadcast cycle length is 25 data items. Consistent with the results in Figure 12 and 13, the performance of SAVI is still the best but the drop of  $D_R$  due to larger drop period is greater as shown in Figure 17. It is because the broadcast cycle length is now shorter and the waiting time for “hot” data items is shorter.

#### 5.4.4 Impact of Cache Size

An important purpose of the AVI approaches is to increase the cache hit probability. Thus, their performance should be highly affected by the size of the cache at the mobile clients. Figure 18 and figure 19 show the values of  $R_R$  and  $D_R$  at different cache sizes respectively. Consistent with our expectation, the performance of the three approaches are about the same when the cache size is

reduced to zero since all of them use the same method, EWMA, to calculate the access scores of the data items for broadcast.

As can be seen in Figures 18 and 19, when the cache is very small, the performance of the system is improved with an increase in cache size due to greater cache hit rate as shown in Figure 20. When the cache is sufficiently large, further increase in the cache size does not improve the system performance as most of the requests can find their data items in the client caches which are large enough for all the “hot” data items. Although increasing the cache size after 50 data items can put more cold data item in the cache, the improvement is not significant as the probability of accessing cold data items is very low. Furthermore, by the time the cold data items are accessed again the cached versions are probably no longer invalid and hence not useful in reducing access delay. Consistent with the results in previous sub-sections, the performance of SAVI is still the best, due once again to lower cache invalid rate as shown in Figure 21.

## 6 Conclusions

The importance of mobile computing systems has resulted in much research in this area recently. An important scenario which has not received adequate treatment in the literature is the case where the mobile clients requests real-time information from the information server. Although the broadcast approach is a good choice as it can utilize the limited bandwidth more effectively, we have shown in this paper that new strategies need to be devised when the temporal constraints on the validity of the data are considered. We developed a data model which considers the temporal properties of the data items explicitly, and designed new data selection algorithms using the concept of Absolute Validity Interval based on how the users use the data items, their update periods and their dynamic properties. In the AVI-based approaches, the selection of data items for broadcast to satisfy the requests from the mobile clients considers both the access frequencies of the data items and their AVI. The main purpose of including the AVI of the data items in selecting the data items is to increase the cache hit probability and thus reduce the access delay for the data items. At the same time, it can eliminate the need for expensive cache invalidation messages to invalidate the cached data items whenever the value of the data items have been updated in the database server.

Two variants of AVI-based approaches are proposed: static and dynamic. Simulation results show that both approaches outperform schemes based purely on the access frequencies of the data items. SAVI performs somewhat better than DAVI as it can maintain the validity of the data items in the client cache more effectively. However, for the special case where the broadcast cycle is very short, DAVI may perform better. We trust that these results will be useful for designers of actual systems particular where the data items exhibit the temporal properties discussed in this paper.

## References

- Acharya, S. 1997. Broadcast Disks: Dissemination-based Data Management for Asymmetric Communication Environments. *Technical Report CS-97-15*, Department of Computer Science, Brown University.
- Acharya, S., Franklin, M., Zdonik, S. 1996. Dissemination Updates on Broadcast Disks. *Proceedings of 22<sup>nd</sup> VLDB Conference*, India.
- Acharya, S., Franklin, M., Zdonik, S. 1997. Balancing Push and Pull for Data Broadcast. *Proceedings of ACM SIGMOD*, Tucson, Arizona.
- Mesquite. 1997. *CSIM 18 User Guide*, Mesquite Software, Inc. (URL: [http://www.mesquite.com/1\\_intro.html](http://www.mesquite.com/1_intro.html)).
- Chan, B.Y.L, Si, A. and Leong, H.V. 1998. Cache management for mobile databases: design and valuation. *Proceedings of the 14th International Conference on Data Engineering*, pp. 54-63.
- Datta, A., Celik, A., Kim, J. and VanderMeer, D.E.. 1997. Adaptive Broadcast Protocol to Support Power Conservant Retrieval by Mobile Users. *Proceedings of 13th International Conference on Data Engineering*.
- Fernandez J. and K. Ramamritham 1999. Adaptive Dissemination of Data in Time-Critical Asymmetric Communication Environments. *Proceedings of 11th Euromicro Conference on Real-Time Systems*, York, England.
- Franklin, M. and S. Zdonik. 1998. Data in Your Face: Push Technology in Prospective. *Proceedings of 1998 ACM SIGMOD Conference*, Seattle.
- Hu, Q. and Lee, D.L. 1997. Adaptive Cache Invalidation Methods in Mobile Environments. *Proceedings 6th IEEE Intl. Symposium on High Performance Distributed Computing Environments*.
- Hu, Q., D.L. Lee and W.C. Lee. 1998. A Comparison of Indexing Methods for Data Broadcast on the Air. *Proceedings in 12<sup>th</sup> International Conference on Information Networking*.
- Hameed, S. and Vaidya, N.H.. 1997. Efficient Algorithms for Scheduling Single and Multiple Channel Data Broadcast. *Technical Report 97-002*, Department of Computer Science, Texas, A&M University.
- Imielinski, T. and Badrinath, B.R. 1994. Mobile Wireless Computing: Challenges in Data Management. *Communications of the ACM*, vol. 37, no. 10.
- Imielinski, T., Viswanathan, S. and Badrinath, B. R. 1994. Energy Efficient Indexing on Air. *Proceedings of ACM SIGMOD Conference*.
- Imielinski, T., Viswanathan, S. and Badrinath, B. R. 1997. Data on Air: Organization and Access. *IEEE Transactions on Data and Knowledge Engineering*, vol. 9, no. 3, pp. 353-372.
- Jing, J., Elmargamid, A., Helal, A., Alonso, F. 1994. Bit-Sequences: an Adaptive Cache Invalidation Method in Mobile Client/Server Environments. *Technical Report CSD-TR-94-074*, Computer Sciences Department, Purdue University.
- Jiang S. and Vaidya, N.H., 1998. Scheduling Algorithms for a Data Broadcast System: Minimizing Variance of the Response Time. *Technical Report 98-005*, Department of Computer Science, Texas A& M University.
- Kuo T.W. and A.K. Mok. 1992. Application Semantics and Concurrency Control of Real-Time Data-Intensive Applications. *Proceedings of IEEE 13th Real-Time Systems Symposium*.
- Kuo T.W. and A.K. Mok. 1993. SSP: a Semantics-Based Protocol for Real-Time Data Access. *Proceedings of IEEE 14th Real-Time Systems Symposium*.
- Kayan E. and Ozgur Ulusoy. 1999. Real-Time Transaction Management in Mobile Computing Systems. *Proceedings of 6th International Conference on Database Systems for Advanced Applications*, Taiwan.

- Lee, W.C., Hu, Q. and Lee, D.L. 1997. Channel Allocation Methods for Data Dissemination in Mobile Computing Environments. *Proceedings of 6th IEEE Intl. Symposium on High Performance Distributed Computing Environments*.
- Leong, H.V. and Si, A. 1995. Data broadcasting strategies over multiple unreliable wireless channels. *Proceedings of the 4th International Conference on Information and Knowledge Management*, pp. 96-104.
- Leong, H.V., Si, A., Chan, B.Y.L. 1996. Caching Data over a Broadcast Channel. *Mobile Communications*, ed. J.L. Encarnacao and J.M. Rabaey, Chapman and Hall.
- Leong, H.V. and Si, A. 1997. Database Caching over the Air-Storage. *The Computer Journal*, vol. 40, no. 7, pages 401-415.
- Pitoura, E. and Bhargava, B. 1993. Dealing with Mobility: Issues and Research Challenges. *Technical Report*, Purdue University.
- Pitoura, E. and Samaras, G. 1997. *Data Management for Mobile Computing*, Kluwer Academic Publishers.
- Ramamrithan, K. 1993. Real-time Databases. *Parallel and Distributed Databases*, vol. 2, no. 1, pp. 199-226, 1993.
- Stathatos, K., Roussopoulos, N. and Baras, J.S. 1997. Adaptive Data Broadcast in Hybrid Networks. *Proceedings of 23<sup>rd</sup> VLDB Conference*, Athens, Greece.
- Su, C. J., Tassiulas, L. 1997. Broadcast Scheduling for Information Distribution. *Proceedings of IEEE Infocom*, Kobe.
- Shivakumar, N. and Venkatasubramanian, S. 1996. Efficient Indexing for Broadcast Based Wireless Systems. *Mobile Networks and Applications*, vol. 1 no. 3.
- Ulusoy, O. 1998. Real-Time Data Management for Mobile Computing. *Proceedings of International Workshop on Issues and Applications of Database Technology (IADT'98)*, Berlin, Germany.
- Vaidya, H. and Hameed S. 1996. Improved Algorithms for Scheduling Data Broadcast. *Technical Report 96-029*, Department of Computer Science, Texas A& M University.
- Wu, K.L., M.S., Yu and Chen, M.S. 1996. Energy-Efficient Caching for Wireless Mobile Computing. *Proceedings of 12th Intl. Conference on Data Engineering*.
- Xuan, P., O. Gonzalez, J. Fernandez & Ramamritham, K. 1997. Broadcast on Demand: Efficient and Timely Dissemination of Data in Mobile Environments. *Proceedings of 3<sup>rd</sup> IEEE Real-Time Technology Application Symposium*.

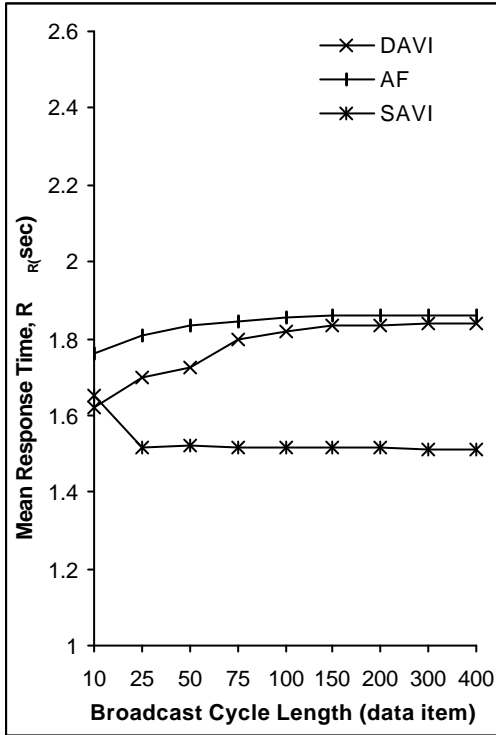


Figure 2 Impact of Broadcast Cycle Length on Mean Response Time

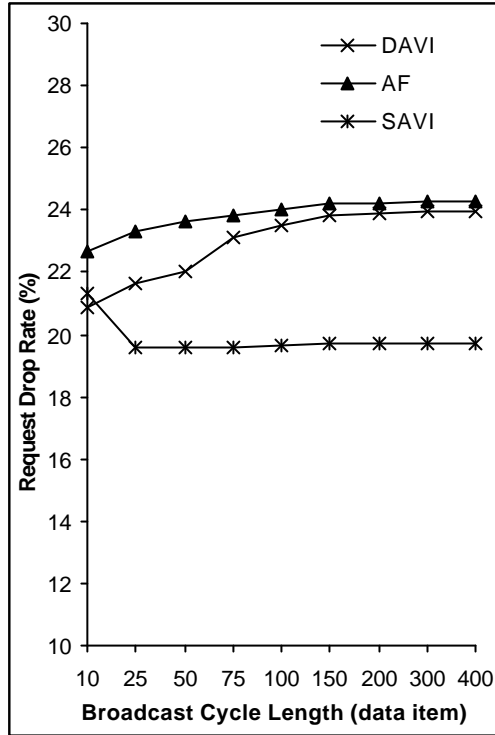


Figure 3 Impact of Broadcast Cycle Length on Drop Rate

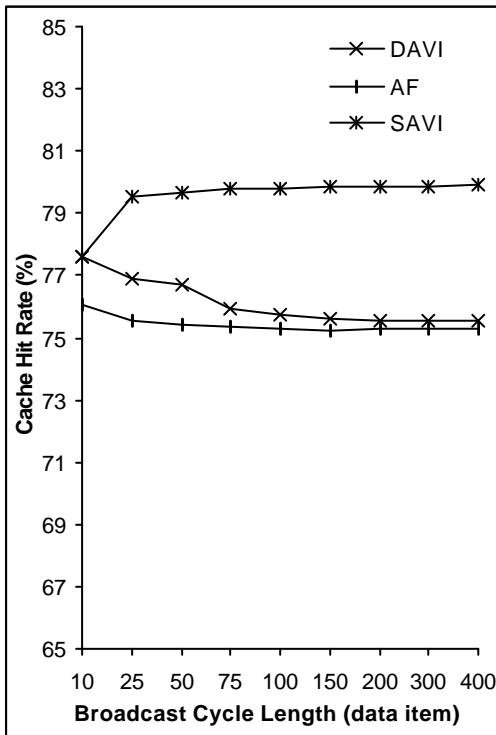


Figure 4 Impact of Broadcast Cycle Length on Cache Hit Rate

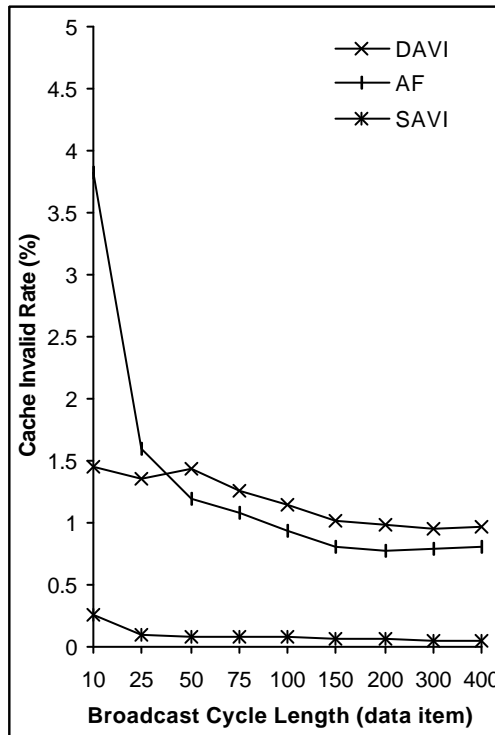


Figure 5 Impact of Broadcast Cycle Length on Cache Invalid Rate

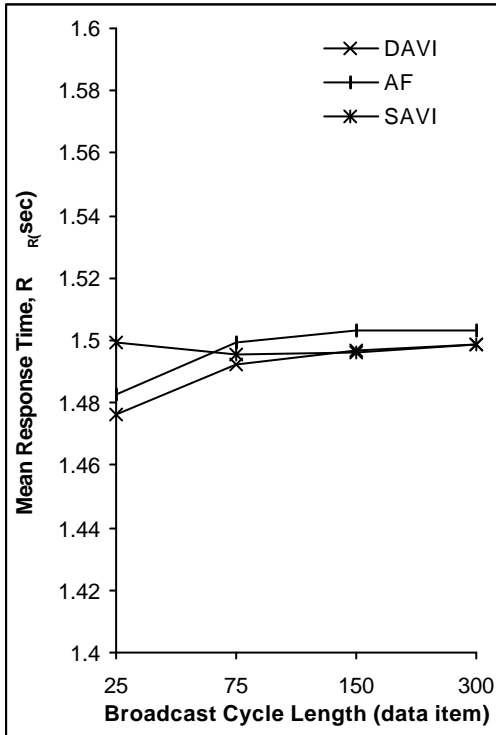


Figure 6 Impact of Broadcast Cycle Length on Mean Response Time

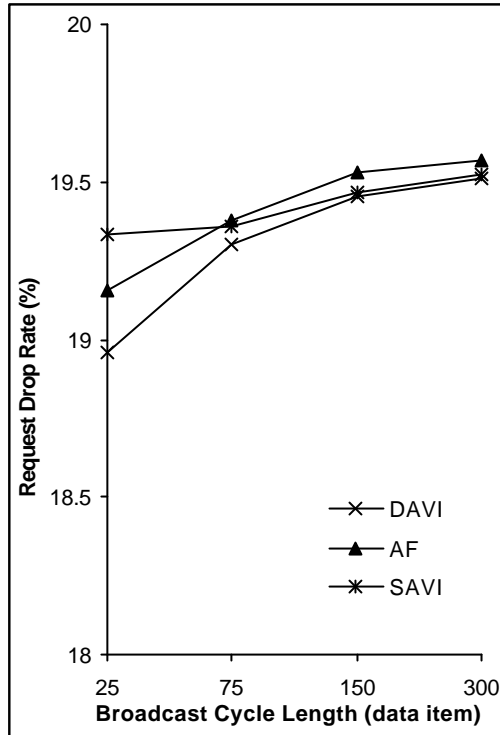


Figure 7 Impact of Broadcast Cycle Length on Drop Rate

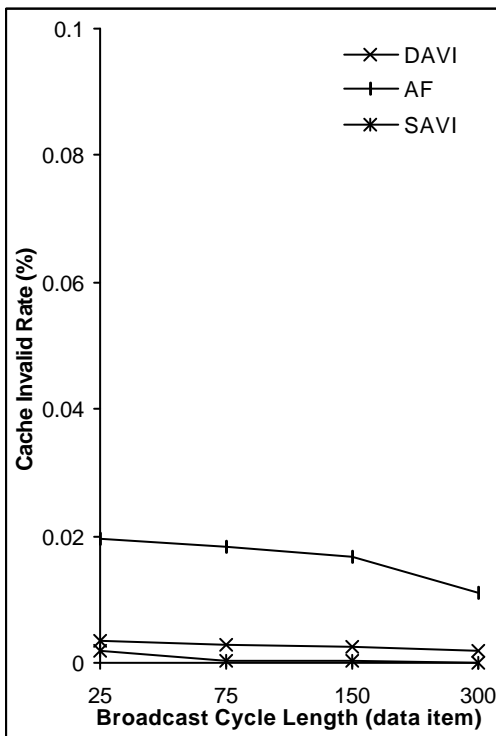


Figure 8 Impact of Broadcast Cycle Length on Cache Invalidation Rate

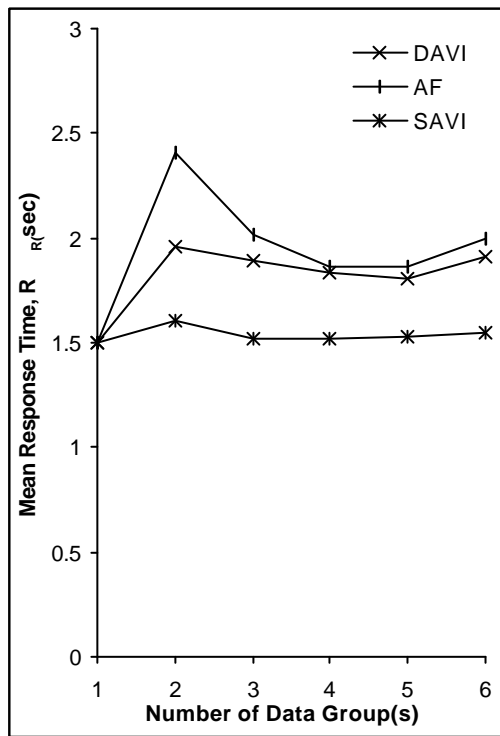


Figure 9 Impact of Number of Data Groups on Mean Response Time

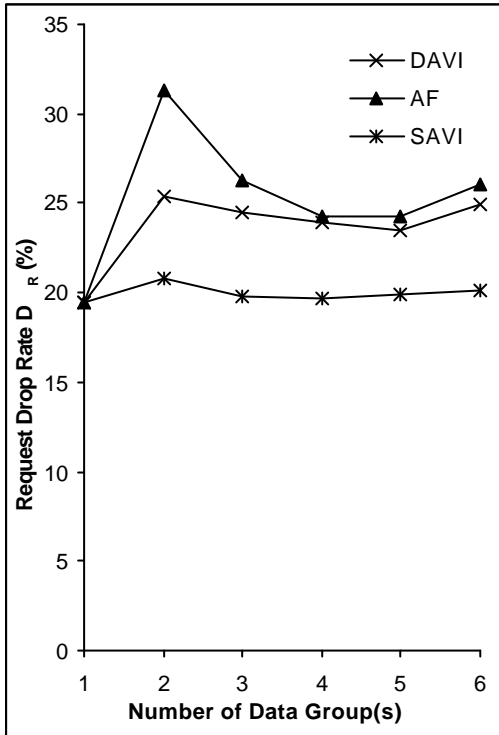


Figure 10 Impact of Number of Data Groups on Request Drop Rate

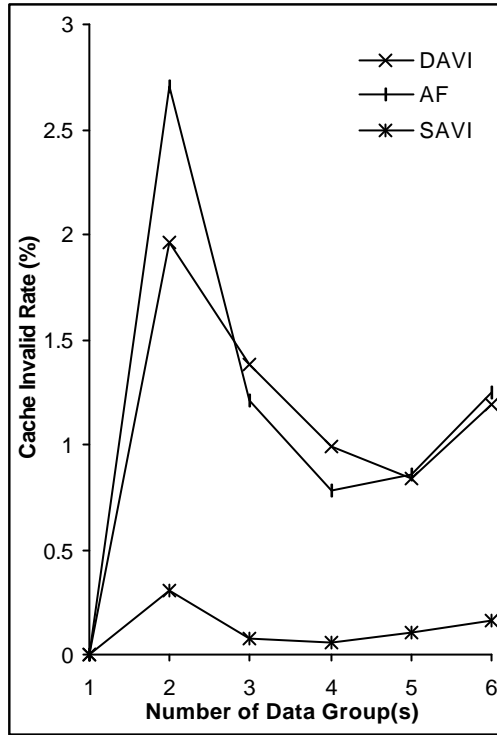


Figure 11 Impact of Number of Data Groups on Cache Invalid Rate

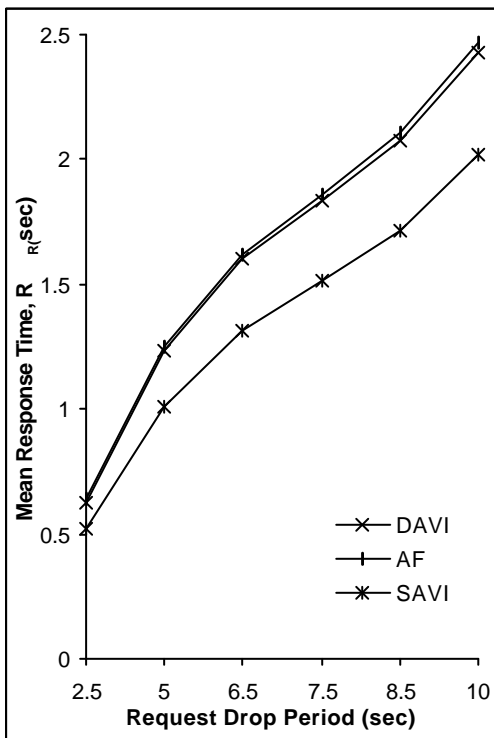


Figure 12 Impact of Request Drop Period on Mean Response Time

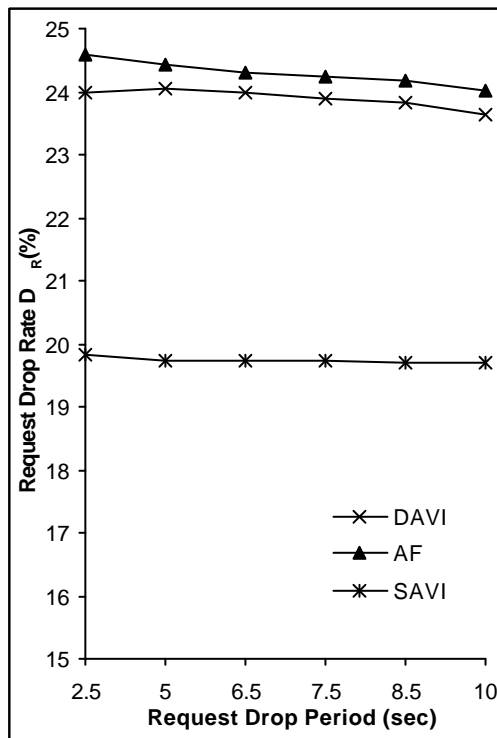


Figure 13 Impact of Request Drop Period on Drop Rate

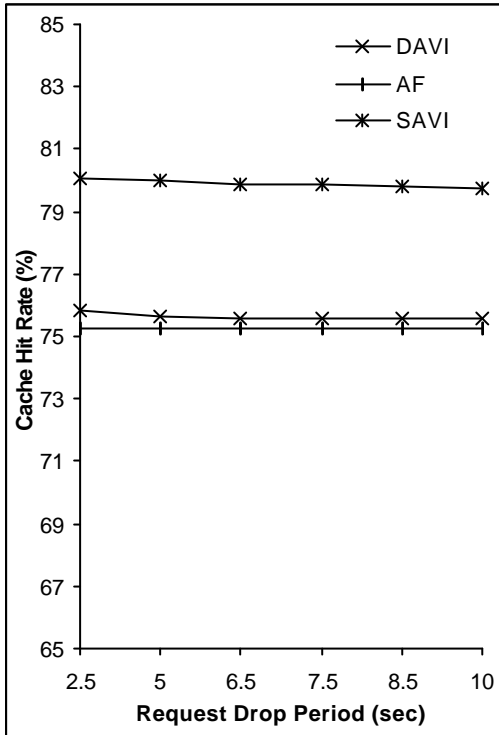


Figure 14 Impact of Request Drop Period on Cache Hit Rate

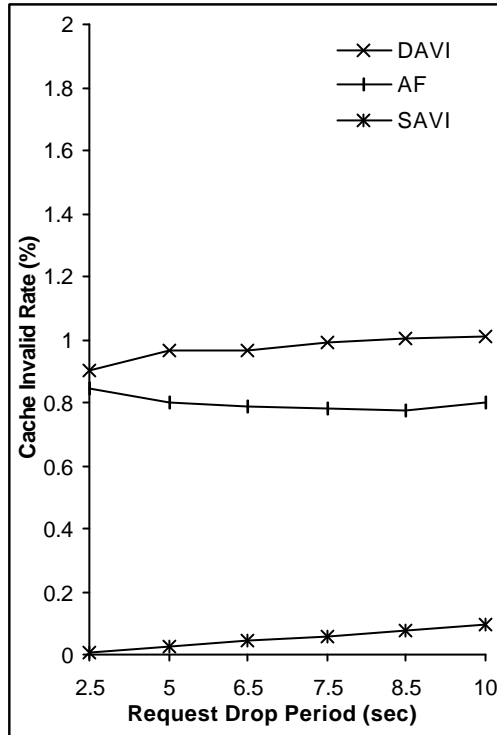


Figure 15 Impact of Request Drop Period on Cache Invalid Rate

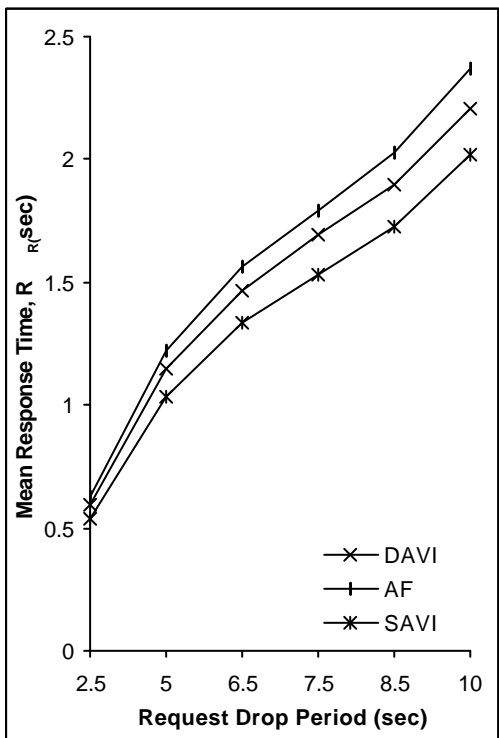


Figure 16 Impact of Request Drop Period on Mean Response Time

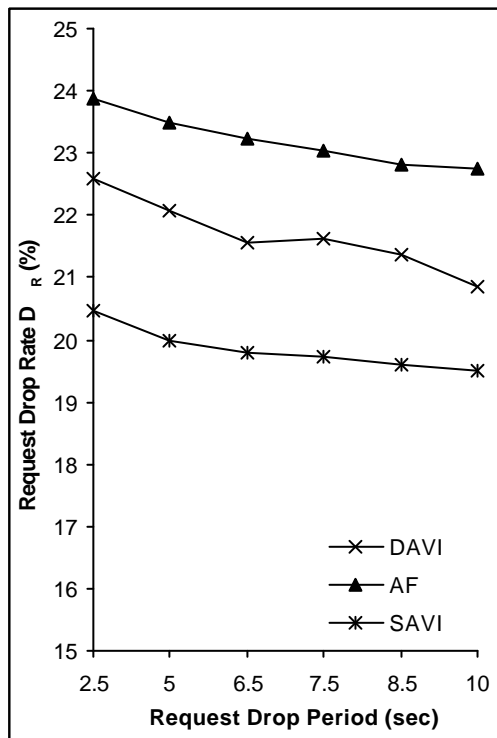


Figure 17 Impact of Request Drop Period on Drop Rate

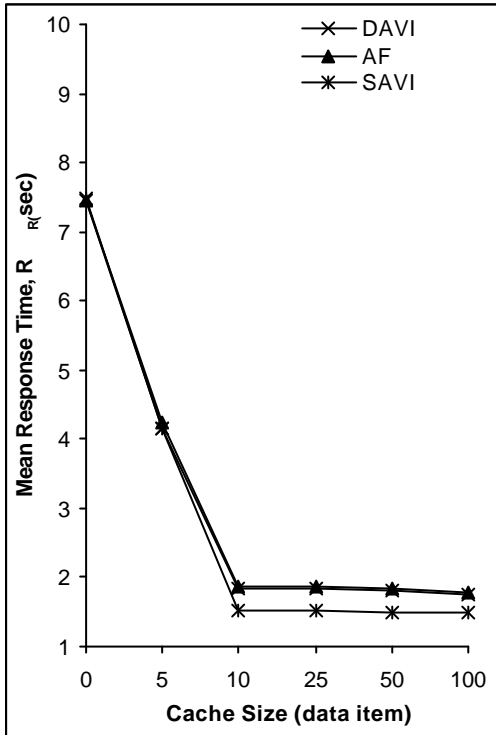


Figure 18 Impact of Cache Size on Mean Response Time

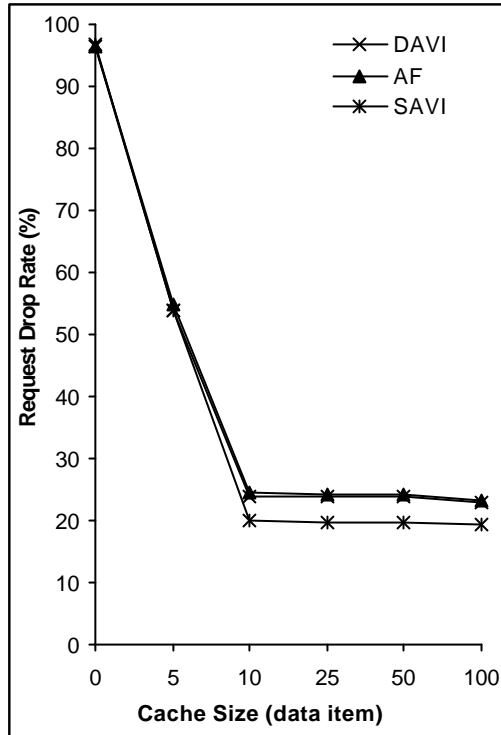


Figure 19 Impact of Cache Size on Drop Rate

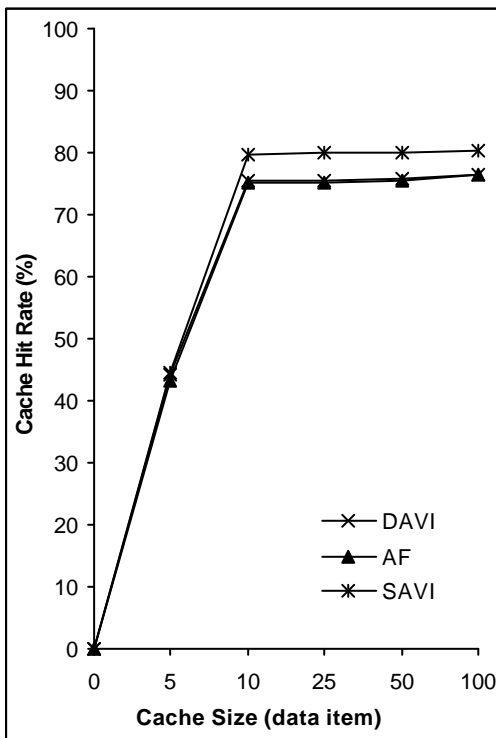


Figure 20 Impact of Cache Size on Cache Hit Rate

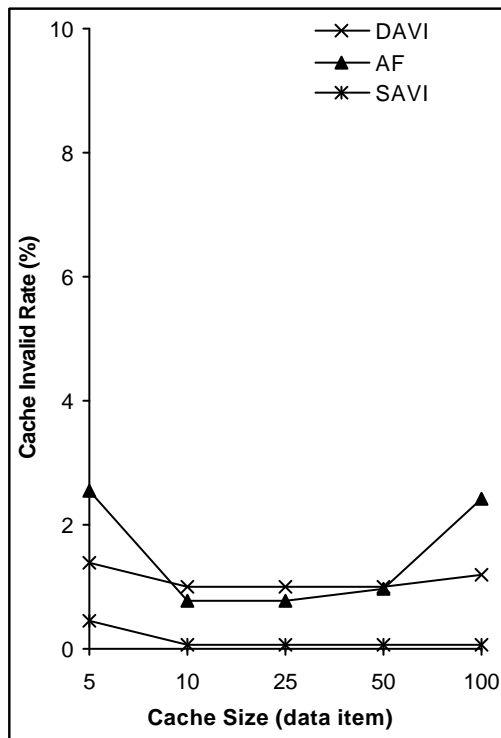


Figure 21 Impact of Cache Size on Cache Invalid Rate