

Broadcast Strategies to Maintain Cached Data for Mobile Computing System¹

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.

Although data broadcast has been shown to be an efficient data dissemination technique for mobile computing systems, many issues such as selection of broadcast data and caching strategies at the clients are still active research areas. In this paper, by examining the dynamic properties of the data items in mobile computing systems, we define the validity of a data item by its absolute validity interval (avi). Based on the avi of the data items, we propose different broadcast algorithms in which the selection of data items for broadcast will be based on the avi of the data items and their access frequencies. The purpose of the AVI algorithms is to increase the client cache hit probability so that the access delay for a data item will be much reduced. Simulation experiments have been conducted to compare the AVI algorithms with the algorithm which only considers the popularity of the data items. The results indicate that the AVI algorithms can significantly improve the mean response time and reduce the deadline missing requests.

1 Introduction

The research in mobile computing systems has received a lot of interest in recent years. A key element in many of these mobile computing systems is the need to distributed real-time information from a database server to mobile clients. Recent research has focused on one of the most important issues: supporting efficient data retrieval and dissemination to the requests from mobile clients [1,2,7,11,12]. 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 requests from mobile clients fall into one of two main approaches: the *on-demand* approach and the *broadcast* approach (also called the *pull* and *push* approach respectively) [1].

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 [1,13]. In the broadcast approach, the information server periodically and continuously broadcasts data items to the mobile clients. If there is a request waiting for a data item, it will get the data item from the "air" while it is being broadcast. Under the broadcast 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. Hence, most of the previous work on data dissemination in mobile systems is based on the broadcast approaches [1,2,4,5,8,9,10,13].

In this paper, we focus on the data dissemination problem using the broadcast approach. An important design issue in this approach is the algorithm used for selecting data items from the database for broadcast. Although a number of algorithms which aims to minimize the

¹ This work was supported in part by City University Strategic Grant #700584.

average delay for data items [1,5, 8,10,13] have been proposed, most of them are based on the popularity, or the access frequencies, of the data items. They have ignored the impact of the timing constraints of the client requests and the temporal properties of the data items on the performance of the algorithms.

In many mobile computing systems, many data items in a mobile computing system are status information, for instance the last traded price of a stock and the current traffic situation of a road. Their values are highly dynamic as they are used to represent the current situations in the external environment. The validity of the current values of the data items is associated with temporal constraints. They may become out-dated and useless with the passage of time. The dynamic properties of the data items can greatly affect the management of data items at the client caches. Maintaining data items at the client caches is critical to the performance of a mobile computing system. If a request can find its required data items at the client cache, the request can be served immediately and its response time can be greatly reduced. Otherwise, it has to wait until the server broadcast the data items. However, the data items at the client cache will become out-dated. A request still has to wait for its data items even though it can find the data items at the client cache but they are out-dated.

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 the this temporal data model, we propose new broadcast algorithms, we call them AVI algorithms, in which the temporal properties of the data items will also be considered to select the data items for broadcast. The purpose of the AVI algorithms is to maintain the data items at the client caches so that the cache hit probability can be improved and consequently, the access delay for the data items can be much reduced.

2 Temporal Data Model for Mobile Computing Systems

When we examine the properties of the data items in a mobile computing system, we can find that many data items in the database are used to record the “real-time” information of the objects in the external environment. Some of the examples are the last traded stock prices, news updates, as well as traffic and weather conditions. Since they track external events, this information may change quite rapidly. In order to maintain the consistency of the data items with the status of their corresponding objects in the external environment, an immediate update approach may be used in which updates are generated to “refresh” the data items in the database whenever the status of the external objects have changed.

Although this approach can provide a tight consistency between the data items in the database and the actual status of the objects in the external environment, tracking every change in the status of the external objects will create a large number of update transactions. For many data items, such as the weather condition, news updates and traffic situations, their updates may be performed periodically. There are two reasons for this design. 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 such as the traffic status of a particular road. When this information is reported to the information, the traffic status of the road will already change to another status. 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 most drivers.

The dynamic properties of the data items make the management of cached data a difficult problem. Whenever a new version of a data item has been created in the database, all the cached version of the data item has to be invalidated. Traditionally, the invalidation is done by sending invalidation messages to the clients which have cached the data item [6]. Due to the delay in invalidation as a result of the slow bandwidth, maintaining a tight consistency between the data items at the client cache and the database is not easy. When a request finds that its required data item is at the client cache, it is not able to determine whether this is the most updated version without incurring heavy overhead.

In order to resolve the problem in accessing the data items at the client cache and to improve the probability of cache hit probability, we introduce a new temporal model to characterize the temporal properties of the data items in mobile computing systems. In the temporal data model, it is assumed that the data items are classified into different *data groups* based on their temporal properties, e.g., their values will change in similar rate. Within each data group, the data items are assigned a time frame, called *absolute validity interval (avi)*, to characterize their temporal properties.

Different types of data items may have different *avi* values. The definition of the *avi* for a data group can be based on how the users will use the data items in that group or their update periods. For example, the generation of news report is every 30 minutes. Thus, the *avi* for the news reports can be defined as 30 min. For some information, which update period may not be fixed, i.e., stock data, their *avi* can be defined based on the user requirements. For example, the stock data will be defined with an *avi* of 10 min. as most of the investors will consider the stock information to be out-dated if it is recorded at ten minutes ago. The *avi* for traffic information may be longer as it is not as dynamic as the stock data.

In the temporal data model, formally, each data item can be defined as a 3-tuple:

$$D_i = (V_i, avi_i, TS_i) \quad (1)$$

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 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 associated with the 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 advantage of using *avi* to define the validity of a data item is that no invalidation message will be required to invalidate the data items in the client caches as 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*. Another advantage of the *avi* approach is that it can help us to design the broadcast algorithms so that the validity of the cached data items can be better maintained.

3 AVI Approaches

In this section, we present the AVI approaches in which the *avi* of the data items will be considered explicitly in selecting the data items for broadcast in addition to the access frequencies of the data items. The purpose of considering the *avi* of the data items is to increase the probability of finding valid data items at the client cache of the mobile clients. Under the temporal data model, 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 them in the client cache under the broadcast approach. The AVI approaches consist of two main parts:

- (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.

3.1 Selection of Data items in the AVI Approaches

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 requests waiting for different data items, the server may not be able to satisfy all the requests before their deadlines with the limited bandwidth. Even though, as assumed in many previous work, the server knows the previous data access patterns of the requests and which data items are waiting by the current requests (e.g., by sampling or by asking the mobile clients to periodically generate request 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.

To solve the problem, different approaches have been proposed to select the data items for broadcast. One of the most common approaches is based on the access frequencies of the data items [5,9,10] in which it is assumed that the data requirements of the requests from a mobile client have locality. It is expected that the broadcast of “hot” data items will serve more client requests and these data items are given higher priorities for broadcast.

In the AVI approaches, both the access frequencies and avi of the data items will be considered in selecting the data items.

3.1.1 Access Frequencies (AF)

In this approach, each data item is assigned an access score based on its access frequency. The 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 a method similar to the EWMA as suggested in [10]. The EWMA method uses a weighed moving average to calculate the access scores of the data items. It gives highest weight to the data items currently requested and the weights assigned to the previously requested data items tail off as they become aged. Data items requested in the current cycle has a weight of one; and they age by a factor α ($\alpha < 1$) for each successive cycle. The final access score of a data item is obtained by summing up of all its scores in each broadcast cycle and then normalized through dividing the sum of all the weights used such as:

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

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

3.1.2 Static AVI Approach (SAVI)

Basically, there are two ways to include the avi of data items into the AVI 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. 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. Firstly, the broadcast cycle is 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 . The value of B_i is defined according to

the value of avi_i . It is defined to be proportional to $1/avi_i$. A larger B_i will be assigned to the data group with a smaller avi_i .

$$B_i = DS \times \left(\frac{1}{\sum_{j=1}^k \frac{1}{avi_j}} \right) \times \frac{S_i}{DB} \quad (3)$$

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 . All of them 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 (the value of B) assigned to the data group has been filled up.

In the SAVI, the data items having shorter avi will have a higher chance to be selected for broadcast. The purpose is to maintain their validity in the client caches. Since these items become invalid quicker. It is expected that this approach can improve the client cache hit rate can be improved.

3.1.3 Dynamic AVI Approach

According to the above equation, 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 very high comparing with the data items in the other groups which have a much smaller avi .

In the Dynamic AVI approach, 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 :

$$\text{Score}(D_i) = f(D_i) \times \left(1 + \left(\frac{1}{\sum_{j=1}^k \frac{1}{avi_j}} \right)^w \right) \times \frac{S_i}{DB} \quad (4)$$

where $f(D_i)$ is a function which is 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 $f(D_i)$.

3.2 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. The new requests still can find a valid version of the data item in the client caches if the data item has been cached in the client caches in the previous broadcast cycles. This technique, which we call it *broadcast skipping*, can be exploited to save bandwidth by reducing unnecessary broadcast.

However, if a data item has not been cached at a mobile client, the use of the 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
Number of selected data item increment by one
Endif
Remove the data item from the unselected data item set
End While
End For

```

4 Performance Study

4.1 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.

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.

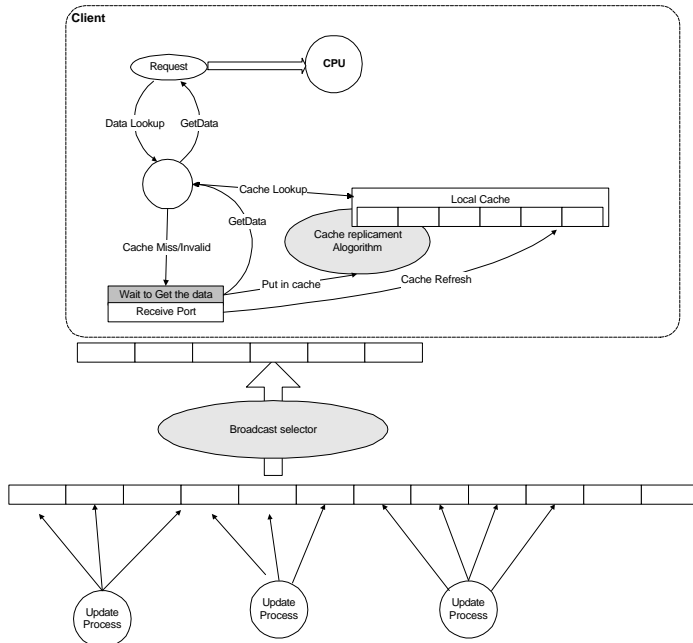


Figure 1. Model of the Mobile Computing System

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

It is assumed that each request requires access to one data item. After the completion of a request, the mobile client will generate the next request after a think time. A client cache is maintained at each mobile client. In processing a request, the cache of the mobile client will be searched first. If the required data item is found in the cache and it is valid, the request will access the data item at the client cache and will be served immediately by the CPU of the mobile client. If the required data item is not in the cache or the data item at client cache is invalid, the request will be blocked until the required data item is being broadcast. After getting the required data item, the request will be processed by the CPU. After the request is served, the data item will also be put in the cache and hence 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 Recent Used (I-LRU) for cache replacement. In I-LRU, the invalid data items will be selected for replacement first. If all the data items are valid, the original LRU principle will be used to select the data items. 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.

It is assumed that the requests from a mobile client 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.

4.2 Model Parameters

The following lists the model parameters and their baseline values:

Parameters	Baseline values
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, α , for calculating access frequency	0.5
w, weighting factor for the DAVI	1
Proportion of hot data items	0.001
Access probability	0.8
Drop period	7.5 sec

4.3 Performance Measures

We use the following four performance measures in our experiments:

- R_R : mean response time of a request
- D_R : dropped rate
- C_H : cache hit rate
- C_I : cache invalid rate

D_R is defined as the number of dropped requests divided by the total number of requests generated. C_H is defined as the number of requests, which requesting 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, which 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.

4.4 Results and Discussions

In this section, we report the performance of the avi approaches. Our simulation program is implemented in CSIM [3]. Due to the limitation of space, we cannot include all the tested performance results. In the following sub-sections, we only show their performance at difference broadcast cycles, avi values and data groups. We have tested the performance of the avi approaches under different deadline constraints of the requests and client cache sizes. The results are consistent with those shown in sections 4.4.1 and 4.4.2.

4.4.1 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 the 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% (e.g., R_R for AF is around 1.85 sec and R_R for SAVI is around 1.5 sec.) smaller than that for AF. D_R for the SAVI is also consistently about 4% smaller than that for the AF. Although the performance of DSAVI 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 DAVI and SAVI, due to the additional consideration given to the avi of the data items in selecting the data items for broadcast, the data items with smaller avi values will have higher chances to broadcast 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 the SAVI is almost zero. Although the cache invalid rate for AF is also very small, however, a few percent decrease in 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 including the factor of the avi of the data items in the broadcast selection approaches can significantly improve the overall system performance.

When we compare the performance of SAVI with DAVI as in Figures 2 to 6, 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 more 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 is smaller when the broadcast cycle length is increases.

4.4.2 Impact of avi Values and Number of Data Groups

In this set of experiments, we increase the avi of the four data groups to be 700sec, 1400sec, 2800sec and 5600sec. 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 large, the cache invalid problem is less serious and the probability of finding an invalid data item at the client cache is low. Thus, C_1 for the three broadcast approaches are about the same and approach zero as shown in Figure 8. C_1 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 4.4.1, the performance of the AVI approaches is better than the AF and SAVI gives the best performance as shown in Figure 9 and Figure 10. Again, the better performance of SAVI is due to low cache invalid rate as shown in Figure 11. C_1 for SAVI remains zero but C_1 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 having a small avi value. Thus, the cache invalid problem becomes more serious.

5 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 has been shown to be 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 have developed a data model which considers the temporal properties of the data items explicitly, and have 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 these approaches, the selection of data items for broadcast to satisfy the requests from the mobile clients consider both the access frequencies of the data items and their avi which characterize their validity. The 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 requirement of invalidation message to invalidate the cache data items whenever the value of the data items have been updated in the database server.

Two AVI approaches are proposed: static and dynamic. The simulation results show that by incorporating the avi of the data items in the selection of data for broadcast with the dynamic and static approaches, the system performance is significantly better than just considering the access frequencies of the data items. SAVI gives a better performance comparing with DAVI as it can maintain the validity of the data items in the client cache in a better way. However, if the broadcast cycle is very small, DAVI may be able to give a better performance.

References

1. Acharya, S., Franklin, M., Zdonik, S., "Disseminating Updates on Broadcast Disks, in Proceedings of 22nd VLDB Conference, India, 1996.
2. Acharya, S., Franklin, M., Zdonik, S., "Balancing Push and Pull for Data Broadcast", in *Proceedings of ACM SIGMOD*, Tucson, Arizona, May 1997.
3. *CSIM 18 User Guide*, Mesquite Software, Inc. (URL: http://www.mesquite.com/1_intro.html).
4. Chan, B.Y.L, Si, A. and Leong, H.V, "Cache management for mobile databases: design and valuation", in *Proceedings of the 14th International Conference on Data Engineering*, pages 54-63, February 1998.
5. Datta, A., Celik, A., Kim, J. and VanderMeer, D.E., "Adaptive Broadcast Protocol to Support Power Conservant Retrieval by Mobile Users", in *Proceedings of 13th International Conference on Data Engineering*, 1997.
6. Hu, Q. and Lee, D.L., "Adaptive Cache Invalidation Methods in Mobile Environments", *Proc. 6th IEEE Intl. Symposium on High Performance Distributed Computing Environments*, 1997.
7. Imielinski, T. and Badrinath, B.R., "Mobile Wireless Computing: Challenges in Data Management," *Communications of the ACM*, vol. 37, no. 10, Oct. 1994.
8. Leong, H.V. and Si, A., "Data broadcasting strategies over multiple unreliable wireless channels", in *Proceedings of the 4th International Conference on Information and Knowledge Management*, pages 96-104, November 1995.
9. Leong, H.V., Si, A., Chan, B.Y.L., "Caching Data over a Broadcast Channel", in *Mobile Communications*, ed. J.L. Encarnacao and J.M. Rabaey, Chapman and Hall, 1996.
10. Leong, H.V. and Si, A., "Database Caching over the Air-Storage", *The Computer Journal*. Volume 40, number 7, pages 401-415, 1997.
11. Pitoura, E. and Bhargava, B. "Dealing with Mobility: Issues and Research Challenges," *Technical Report*, Purdue Univ., Nov. 1993.
12. Pitoura, E. and Samaras, G., *Data Management for Mobile Computing*, Kluwer Academic Publishers, 1997.
13. Xuan, P., O. Gonzalez, J. Fernandez & Ramamritham, K., "Broadcast on Demand: Efficient and Timely Dissemination of Data in Mobile Environments", in *Proceedings of 3rd IEEE Real-Time Technology Application Symposium*, 1997.

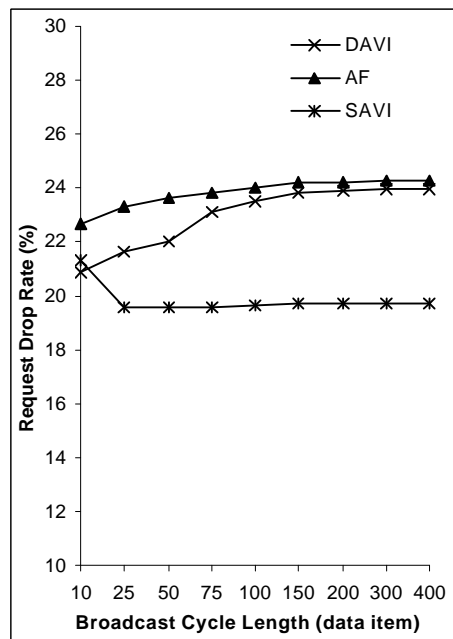
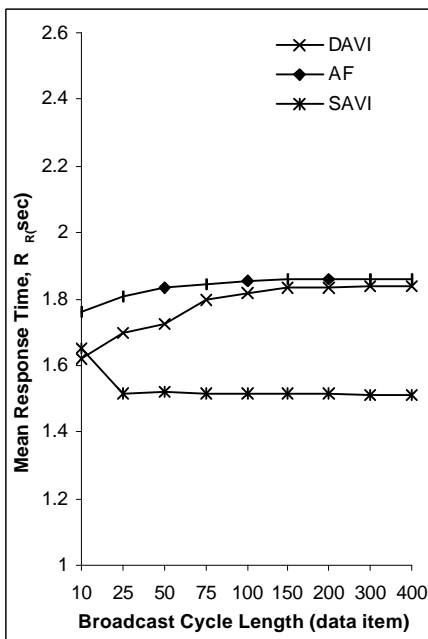


Figure 2. Impact of Broadcast Cycle Length

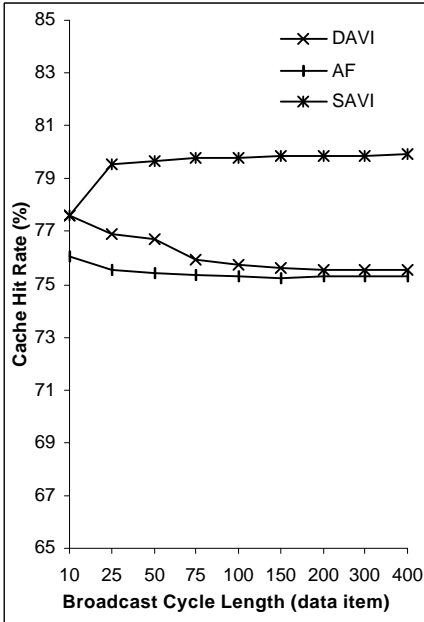


Figure 3. Impact of Broadcast Cycle Length on

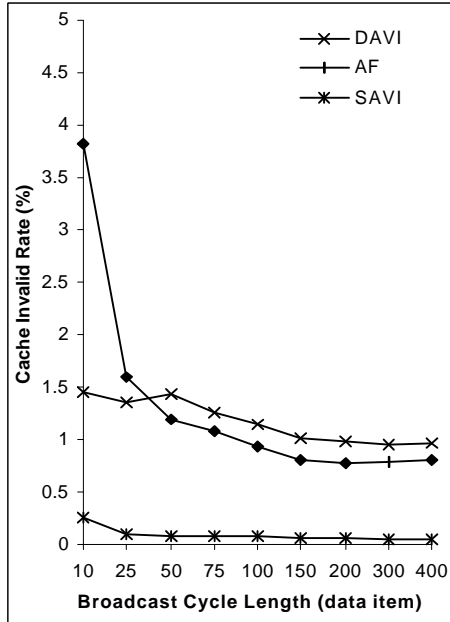


Figure 5. Impact of Broadcast Cycle Length on Cache Invalid Rate

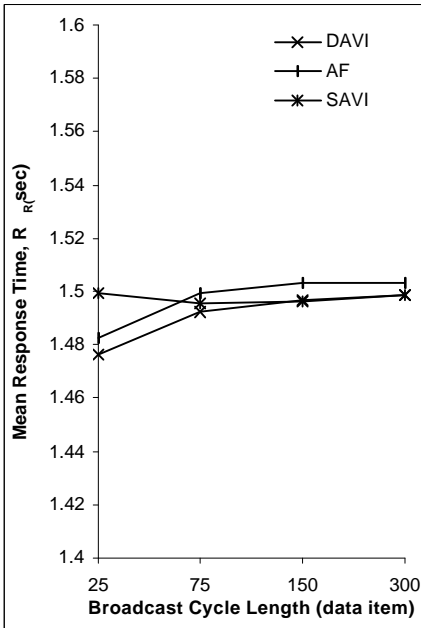


Figure 6. Impact of Broadcast Cycle Length

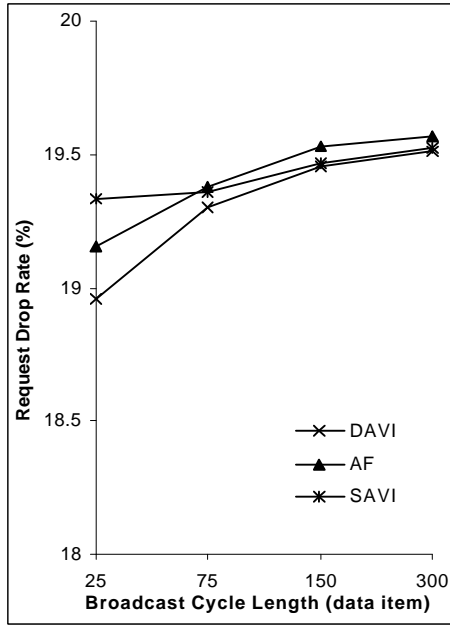


Figure 7. Impact of Broadcast Cycle Length on

on Mean Response Time

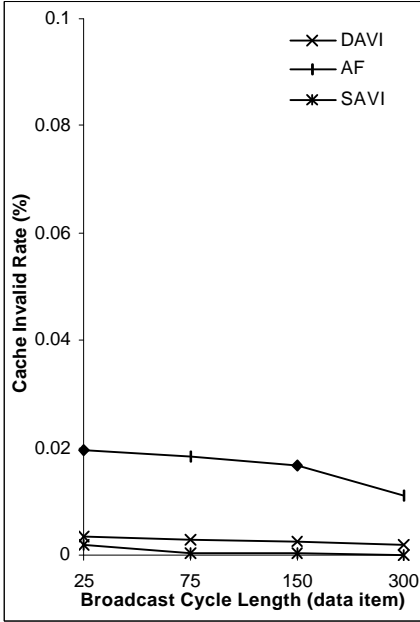


Figure 8. Impact of Broadcast Cycle Length on Cache Invalid Rate

Drop Rate

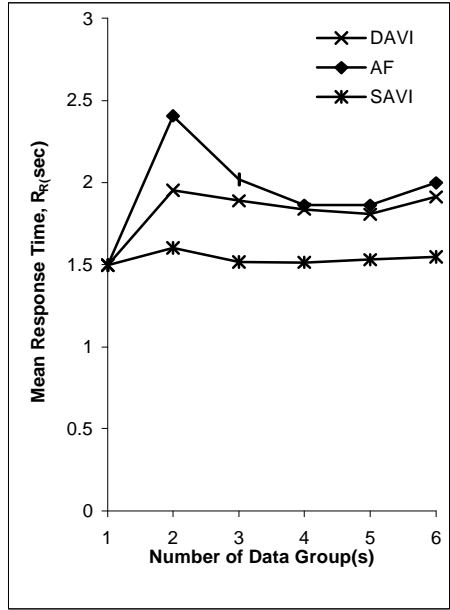


Figure 9. Impact of Number of Data Groups on Mean Response Time

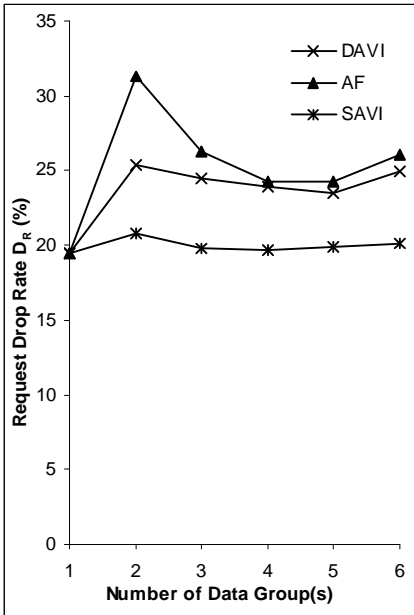


Figure 10. Impact of Number of Data Groups on Request Drop Rate

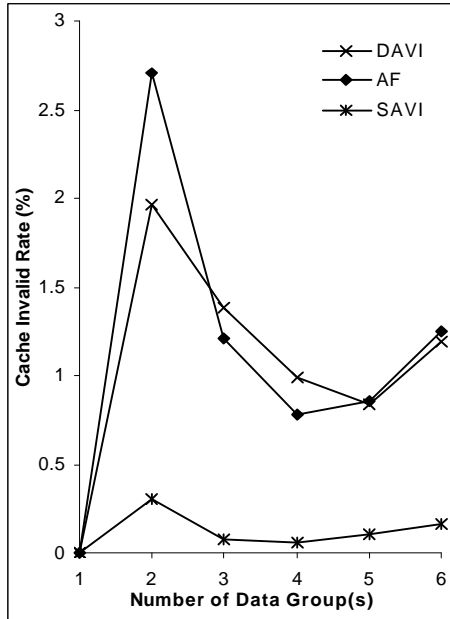


Figure 11. Impact of Number of Data Groups on Cache Invalid Rate

