

# Data Broadcast for Time-Constrained Read-Only Transactions in Mobile Computing Systems<sup>1</sup>

Kam-yiu Lam, Edward Chan and Chun-Hung Yuen  
Department of Computer Science  
City University of Hong Kong  
83 Tat Chee Avenue, Kowloon  
HONG KONG

Email: { cskylam|csedchan|csjcyuen@cityu.edu.hk }

## Abstract

*In this paper, we propose broadcast algorithms, based on the policies proposed in previous studies, to disseminate data items to time-constrained read-only transactions. In the proposed algorithms, the number of data items requested by a transaction will be considered explicitly in selecting the data items for broadcast. The purpose of the algorithms is to reduce the number of transactions missing their deadlines, and to minimize their mean response time. A detailed simulation model of a mobile computing system has been developed and extensive simulation experiments have been performed to study the performance characteristics of the proposed broadcast algorithms, and the results show that our broadcast algorithms outperform previously proposed algorithms.*

Keywords: mobile computing, data broadcast, time-constrained transactions, data dissemination

## 1 Introduction

With advances in mobile communication technologies and mobile computers, research on mobile computing systems has received growing interests in recent years. An important application of mobile computing systems is to support the dissemination of a large amount of information to a large number of mobile clients such as in such as real-time traffic information and real-time stock monitoring systems. In real-time stock monitoring systems, investors receive real-time stock updates from investment companies or data vendors through the web.

In a mobile computing system, mobile clients generate transactions to retrieve various types of information from the information server such as weather broadcast, stock data and news updates. These transactions are *soft* real-time transactions as they are associated with deadlines on their completion times. The deadlines are either defined explicitly by the mobile clients or implicitly defined as a part of the system requirement. If the required information is not retrieved before the transaction deadline, the usefulness of the information will be greatly affected [14]. Thus, the information server is a real-time database which maintains real-time information of the external environment such as the last traded prices of the stock and current status of the roads. Update transactions are created (periodically or sporadically) by sensors, which capture the status of the external environment, and subsequently transmitted the server to maintain the external consistency of the database at the information server.

Due to the intrinsic constraints of mobile computing systems such as limited bandwidth, limited electric power supply [8, 16] and unreliable communication, the design of an efficient and cost effective mobile computing system requires solution to many new problems, which need not be considered in wired distributed systems [7, 11]. One of the most important issues is efficient data dissemination to transactions from mobile clients [4]. The most common method for data dissemination is data broadcast [1]. In the broadcast approach, the information server periodically and continuously broadcasts data items one by one to the mobile clients. If there is a transaction waiting for a data item, it will get the data item from the being broadcast. Thus, the cost for data dissemination is independent of client number since a data broadcast can

---

<sup>1</sup> This work was supported in part by City University of Hong Kong Strategic Grant number 7000584.

satisfy multiple transactions for the same data item, resulting in a much more efficient way of using the bandwidth [1, 4].

An important issue in data broadcast is the selection of data items to broadcast. [7]. A number of algorithms have been proposed in recent years, most of which are based on the access frequencies of the data items and the deadlines of the transactions [2, 9, 3, 15, 17]. However, all these algorithms assume implicitly that each transaction only accesses one data item. This is not realistic for many real-life applications. For example in a stock price request, an investor may need the last traded prices of several stocks at the same time, or a doctor may request the medical history of a patient for a period of time and this information may be stored in different database in the server.

The proposed broadcast algorithms may not work well when the transactions may access more than one data item. It is possible that some of the transactions may miss their deadlines due to the long waiting time for some of their requested data items. In this paper, we design broadcast algorithms for transactions with multiple data requests, with the assumption that all the transactions are read-only. We will examine the proposed broadcast algorithms in the previous work. Based on the principles used in these algorithms, two new broadcast algorithms, called *Request Proportion (RP)* and *Equal Slack Data (EQSD)*, are proposed.<sup>2</sup>

The organization of the remaining parts of the paper is as follows. Section 2 reviews the related work on the topic. Section 3 introduces our mobile computing system model. Section 4 introduces our algorithms to select data items for broadcast based on the number of data items requested by a transaction. Section 5 studies the performance of the proposed broadcast algorithms. The conclusions of the paper are in Section 6.

## 2 Related Work

The design of mobile computing systems has received a lot of attention in recent years [7]. Previous studies have shown that the data broadcast approach can better utilize the limited bandwidth of a mobile computing system and is particularly suitable for the dissemination of large volume of data to a large number of mobile clients at the same time [1, 17]. A number of recent papers consider the broadcast selection issue explicitly [6, 9, 12, 13]. The most common methods are based on the access frequencies of the data items. An example is the *Exponential Weighted Moving Average (EWMA)* approach

---

<sup>2</sup> A related problem is data inconsistency due to database update while the data item is being broadcast. Interested reader may refer to [10] for an efficient concurrency control protocol to resolve the problem.

[9] where requests from the mobile clients are assumed to have locality properties, e.g., for a particular client, certain data items may have higher probabilities to be accessed by the requests from the client than the other data items.

Yet another method is based on the deadline of the transactions. In the Earliest Deadline First (EDF) approach proposed in [17], data items with an earlier deadline will be broadcast first. In order to increase the probability that a data item will be used by more than one data requests, a batch EDF has been suggested in which the broadcast of data items is periodic. In [17], it has been shown that the batch EDF approach performs well if the arrival rate of the data requests is not very high.

## 3 Mobile Computing System Model

A mobile information system consists of an information server and a number of mobile clients connected to the server through a low bandwidth wireless network. In the information server, a real-time database is maintained to keep track of the “real-time” status of the objects in the external environment. The information maintained in the server is “refreshed” by the installation of update transactions which capture the most current status of the objects in the external environment.

As the mobile clients move around, they generate transactions sporadically to retrieve different information from the information server. It is assumed that each transaction is a collection of simple data requests (read operations) and the data requests can be performed in any order. For example, in a stock request transaction, the mobile client may request the last traded price of several stock item, but does not specify which stock item needs to be retrieved first. In the mobile client, there is a cache for maintaining the most recently accessed data items. The cache size is limited due to the relatively small size of the main memory in the mobile clients. It is assumed that whenever an update transaction has been performed in the server database, invalidation messages will be generated to invalidate the cache copies in the client caches [2, 16, 9]. In processing a data request, if the required data item is in the client cache, it may retrieve the data item. Otherwise, the data request has to wait for the broadcast of the data item and get it from the “air”. When all the data requests of a transaction have been served, the transaction is completed. Each transaction is associated with a deadline on its completion time, and missing the deadline renders the transactions meaningless and these transactions will be aborted.

Under the data broadcast approach, the information server broadcasts data items continuously from the database one by one until the end of a broadcast cycle. The length of a broadcast cycle may be fixed or variable. One broadcast cycle is followed immediately by the next cycle, with no intervening gap. The issue of how

what data is to be selected for broadcast is the topic of the next section.

## 4 Selection of Data Items for Broadcast

The primary issue in the design of broadcast algorithm is to determine which transactions should be served first under the limited broadcast bandwidth. The simplest method is of course to use a *sequential method*. The information server starts to broadcast from the first data item until all the data items have been broadcast. Then, it restarts the broadcast cycle from the first data item again. This method is simple and is suitable for the systems having a small database. However, if the database size is large, the waiting time for a data item can be very long and the transactions will have a high probability of missing their deadlines.

In the following sub-sections, we will review the previously proposed broadcast algorithms and discuss their problems when they are applied for disseminating data items to transactions with multiple data requests. Then, we will introduce our broadcast algorithms, the Request Proportion (RP) and Equal Slack Data (EQSD).

### 4.1 Access Frequencies Based Algorithms

The most common approaches are based on the access frequencies of the data items and the deadlines of the transactions. In access frequencies based algorithms [2, 9, 3], it is assumed that the data requirements of the transactions from a mobile client have locality in data accesses. Some of the data items may have higher probabilities to be accessed by the transactions from a mobile client than the other data items. By tracking the access frequencies of the data items from the mobile clients, the system can identify which data items are “hot” data items and which are “cold” data items. It is expected that the broadcast of “hot” data items will have higher probabilities of meeting the data requirements of the transactions.

Various methods have been suggested to calculate the access frequencies of the data items to identify the “hotness” of the data items. A popular method is the EWMA [9] which has been shown to give a good performance. In this method, each data item is assigned an access score to indicate its access frequency. A higher score means that the data item is more popular and is more likely to be requested by the transactions from mobile clients. The data item which is requested in the current broadcast cycle is given a weight of one in calculating the access score. This weight is reduced in every successive cycle by a factor,  $\alpha$ , which amounts to an exponential reduction in the weight and its value is smaller than one. The resulting access score of a data item is obtained by summing up the weights assigned to it and then

normalized through dividing the sum of all the weights used:

$$f_n = (f_n + \alpha f_{n-1} + \alpha^2 f_{n-2} + \dots + \alpha^{n-1} f_1) / S$$

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

It is possible to extend this method to apply to transactions with multiple data requests. However, we have to determine how to calculate the final access score of a data item when more than one transaction wants to access the same data item. The simplest way is to use an *accumulative method*, which means the final access score of a data item is the summation of its scores from all the transactions. The next important question is what score should be assigned to a data item when the transaction wants to access several data items. This is an important issue and can significantly affect system performance. Consider the following example. Initially, the data items  $d_1$  to  $d_5$  have the same score  $s$  based on access frequencies in previous cycles. Now, transactions  $Q_1$ ,  $Q_2$  and  $Q_3$  arrive and their requested data items are as follows:

$Q_1$ :	$d_1$	$d_2$	$d_3$
$Q_2$ :	$d_5$		
$Q_3$ :	$d_2$	$d_4$	

Then, the scores for  $d_1$  to  $d_5$  will be:

$$\begin{aligned} \text{Score}(d_1) &= s + 1 \\ \text{Score}(d_2) &= s + 1 + 1 \\ \text{Score}(d_3) &= s + 1 \\ \text{Score}(d_4) &= s + 1 \\ \text{Score}(d_5) &= s + 1 \end{aligned}$$

Note that the data items requested in the current broadcast cycle will be given a score of 1 as suggested in the EWMA scheme. Thus,  $d_2$  has the highest score and will be given the highest priority for broadcast. As we can see in the above example, the broadcast of  $d_2$  cannot satisfy any of the three transactions.  $Q_1$  still has to wait for  $d_1$  and  $d_3$ , and  $Q_3$  has to wait for  $d_4$ . Thus, some transactions may miss their deadlines even though they have received some data items. The obvious drawback of this approach is that the selection of data items completely ignores the total number of data items requested by a transaction.

In order to solve this problem, we have proposed a new method, called *Request Proportion (RP)*, based on the EWMA for calculating the access scores of data items for transactions with multiple data requests. (Note that RP can also be applied to other broadcast algorithms based on access frequencies, e.g., [3]) In RP, the score assigned to a data item depends on the number of transactions which needs to access that data item, as well as the number of

data items requested by the transactions themselves such as:

$$f_n = (f_n + \alpha f_{n-1} + \alpha^2 f_{n-2} + \dots + \alpha^{n-1} f_1) / S$$

where  $f_i$  is redefined as:

$$f_i = \frac{1}{\sum_j \text{number of data items requested by the transaction } T_j}$$

In RP, if the number of data items requested by a transaction is large, the score assigned to each individual requested data item will be reduced. The purpose is to identify the contribution of a data item to the completion of a transaction. The rationale behind RP is that we aim to complete more transactions before their deadlines instead of simply trying to satisfy more data requests. It is obvious that the transactions with smaller number of data requests will have a higher probability to get their data items since their requested data items are given higher scores. The tradeoff is that the scores of the requested data items of the long transactions may be lower. However, the broadcast of data items for the short transactions at the same time may satisfy the data requests of the long transactions especially if these data items are “hot” data items.

In RP, the score of a data item depends on both its access frequency and the number of data requests of the transactions which have accessed or are waiting for the data item. On the other hand, in EWMA the access score of a data item is solely dependent on its access frequency. If one of the requested data items in a transaction is a “cold” item, the transaction will have a high probability of missing its deadline. This problem is partly solved in RP, because if some short transactions need to access the data item, a higher score will be assigned to the data item even though it is “cold”.

## 4.2 Deadline-Based Algorithms

Another common method for selecting data items for broadcast is based on the deadlines of the requesting transactions, examples include EDF and batch EDF [17]. If there are multiple transactions waiting for the same data item, the transaction with an earlier deadline will be assigned as the deadline of the data item, called *data deadline*. If there is no transaction waiting for a data item, its data deadline will be set to be infinity. The data item with the closest deadline will be assigned the highest priority for broadcast.

Although deadline-based algorithms work well, their performance is questionable if applied directly to transactions with multiple data requests. The deadlines of transactions are not good indicators of their urgency since the number of data items requested by a transaction can be very different. For example, two transactions  $Q_1$  and  $Q_2$

have deadlines 100 and 80 respectively. Suppose they both arrive at time 0.

$$\begin{aligned} Q_1: & \quad d_1, d_2, d_4, d_5, d_6 \\ Q_2: & \quad d_3 \end{aligned}$$

If we apply the EDF method directly, data item  $d_3$  will be assigned a deadline of 80 and the deadlines for data items  $d_1, d_2, d_4, d_5$  and  $d_6$  will be 100. Thus,  $d_3$  will have a higher priority for broadcast. Consequently,  $Q_1$  will have a high probability of deadline missing.

From the above example, it is obvious that if we use the ultimate deadline of a transaction as the basis for assigning data deadlines, the transaction with single data request will be favored. To solve the problem, we need to tighten the deadline of the individual data items, which are requested by the transactions with multiple data requests, in order to allow the longer transactions to have a reasonable chance of being completed within their deadlines. We propose to use a sub-deadline assignment method, called *Equal Slack Data (EQSD)*, in which the data deadlines act as better indicators of the urgency of their requesting transactions. In EQSD, the deadline to be assigned to a data item,  $d_j$ , requesting by a transaction,  $Q_i$ , is defined as:

$$\text{deadline}(d_j) = \text{ar}(Q_i) + \frac{\text{deadline}(Q_i) - \text{ar}(Q_i)}{\text{number of data items requesting by } Q_i}$$

where  $\text{deadline}(Q_i)$  and  $\text{ar}(Q_i)$  are the deadline and arrival time of  $Q_i$ , respectively. Referring to the above example, the deadline of  $d_3$  will be 80 and the deadlines of  $d_1, d_2, d_4, d_5$  and  $d_6$  will be 20. Thus, these data items will be broadcast before  $d_3$ .

By assigning deadlines to data items that can faithfully represent the urgency of the transactions, it is expected that the progress of the transactions can be better monitored and consequently they will have a higher probability of meeting their deadlines. In EQSD, transactions with more data items usually get their requested data items earlier comparing with the transactions with similar deadlines but smaller number of data requests. However, this is not always true. Under the EQSD method, the priority of a data item will become higher even though it is requested by a single data item transaction as the time passed by. Thus, the EQSD attempts to balance the urgency of the transactions and their lengths.

## 5 Performance Study

### 5.1 Simulation Model

Figure 1 depicts the model of the mobile computing system used in our simulation experiments. To

simplify the model, it is assumed that the data items in the database are classified into different data groups. The data items belonging to the same group will have the same update rate which defines the arrival rate of update transactions on the data items in the group. The update transactions are generated by the update processes.

Each transaction generated from a mobile client is defined with a drop period which is a random number uniformly distributed between the upper and lower drop bounds. The deadline of a transaction is defined as its arrival time plus its drop period. If the transaction cannot be completed before its deadline, it will be dropped immediately. A transaction consists of a collection of data requests. It is assumed that each transaction may have one to five data requests uniformly distributed and each data request will access one data item. The data requests from a transaction can be performed in any order.

The transactions from a mobile client have locality in their references to data items. The locality of reference is defined by two parameters, *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 which are evenly distributed in the database.

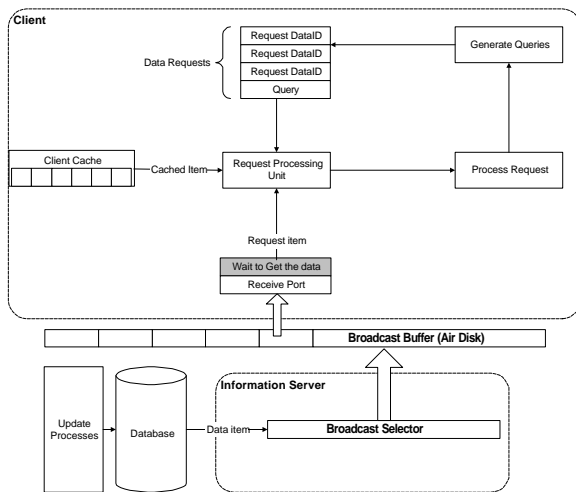


Figure 1: Model of the Mobile Computing System

A cache is maintained in each mobile unit. In processing the data requests of a transaction, the cache of the mobile unit will be searched first. If the required data item is in the cache, the data request will be served immediately by the CPU of the mobile unit. If the required data item is not in the cache, the data request will be blocked until the requested data item is being broadcast. When the requested data item has arrived, the data request will be processed by the CPU. When the processing is completed, a copy of the data item will also be put in the cache and hence available for future transactions. When

all the data requests of a transaction have been served, the transaction is completed. The client will then generate the next transaction after an appropriate think time.

If the cache is full, a cache replacement algorithm will be invoked to free the space for the incoming data item. In our model, a FIFO policy is adopted due to its simplicity. (In our simulation experiments, we have found that the use of different cache replacement algorithms, such as least recently used, does not have any significant effect on the system performance. It may be due to the fact that this is not an important issue on the system performance.)

### 5.1 Model Parameters and Performance Measures

The following is the list of the model parameters and their baseline values:

Parameters	Baseline values
Broadcast cycle length	25 –100 data items
Database size	4000 data items
Number of data item groups	4
Update rate of the group 1 data items	50 sec
Update rate of the group 2 data items	100 sec
Update rate of the group 3 data items	200 sec
Update rate of the group 4 data items	400 sec
Number of mobile clients	200
Broadcast bandwidth	18kbps
Broadcast time for a data item	100 –200 ms (uniformly distributed)
CPU service time for a data request	1.0 –5.0 ms (uniformly distributed)
Think time	0 second
Cache size	10 data items
Weighting factor for EWMA (used in RP & EWMA only)	0.5
Proportion of hot data items	0.01
Access probability of hot data items	0.8
Drop period	35 –55 sec

The bandwidth of the mobile communication link is chosen to be 18.8 kbps (this is the typical capacity of the data channels in cellular mobile network such as PCS and GSM.). Due to the low bandwidth, the size of the data items cannot be large. Otherwise, it will take a long time to broadcast a data item. It is assumed that the sizes of the data items are in the range of 225 to 250 bytes. Thus, the time required to broadcast a data item is in the range of 100 to 200ms. In order to control the workload more

easily, the think time is set to be zero. Under such setup, the workload can be changed by varying the number of mobile clients.

The performance measures used in the simulation experiments are:

- $R_R$ : mean response time of a transaction (including those dropped)
- $D_R$ : dropped rate

$D_R$  is defined as the number of dropped transactions divided by the total number of transactions generated. It measures the capability of the system in meeting the deadlines of the transactions.

## 5.2 Results and Discussions

### 5.2.1 Comparing the Performance of Batch EDF and Batch EQSD

Figure 2 and Figure 3 depict the mean response time ( $R_R$ ) and drop rate ( $D_R$ ), respectively, for batch EDF (bEDF) and batch EQSD (bEQSD) at different broadcast cycle lengths with and without client cache (marked C0 and C10 respectively). In Figure 2, it can be observed that the performance of bEQSD is much better than bEDF.  $R_R$  for bEQSD is consistently about 30% and 20% lower than that for bEDF at C0 and C10 respectively. Consistent with our intuition,  $R_R$  at C10 is lower than at C0 as some of the data requests can find their data items at the client cache which can greatly reduce the delay for getting the data items. The values of  $R_R$  and  $D_R$  are not much affected by broadcast cycle lengths. The reason may be that although increasing the broadcast cycle length can meet the data requirement of more transactions, it also increases the waiting time for the data items. In Figure 3, we can see that at C0,  $D_R$  for bEQSD is much lower than that for bEDF. Thus, when all the transactions have to wait data items from the “air”, the use of bEQSD can significantly improve the overall whole system performance. However, the reverse is true at C10. The reason may be that the transactions with shorter lengths have higher probabilities to be dropped in the bEQSD than in the bEDF.

Figure 4 to Figure 5 show the results when the workload is increased (400 clients). As shown in Figure 4, consistent with the results in Figure 2,  $R_R$  for bEQSD is much lower than that for bEDF at both C0 and C10. Unlike Figure 3,  $D_R$  for bEQSD is better than bEDF at both C0 and C10 as shown in Figure 5. At heavy workload, the long transactions are more likely to miss their deadlines and using bEQSD can greatly improve their performance.

Figure 6 to Figure 9 show the results when the drop bounds are reduced to 20 to 40 sec (implying tighter deadline constraints) and Figure 10 to Figure 13 show the

results when the drop bounds are 50 to 70 sec (looser deadline constraints). As shown in the figures, the performance of bEQSD is always better than bEDF in  $R_R$  and the relative performance of bEQSD and bEDF is roughly the same as those shown in Figure 2 to Figure 7. From the figures (Figures 7, 9, 11 and 13),  $D_R$  for bEQSD is smaller than that for bEDF when the workload is heavy (400 clients) and deadline constraints are tight (drop period = 20 –40). It is because under tighter deadline constraints and heavier workload, long transactions will have higher probabilities of missing their deadlines and using bEQSD can improve their performance.

### 5.2.2 Comparing the Performance of RP and EWMA

Figure 14 and Figure 15 compares the performance of RP and EWMA. As shown in Figure 14,  $R_R$  for RP is consistently lower than that for EWMA under different broadcast cycle lengths. The improvement is particularly noticeable at smaller broadcast cycle especially when there is no client cache. This is because when the broadcast cycle is short, the selection method becomes more critical to the system performance as only a few data items can be broadcast. Although  $D_R$  for RP and EWMA are similar at C10 as shown in Figure 15,  $D_R$  for RP is much lower (about 40%) than EWMA at C0. This means that when all the transactions have to wait for broadcast data, the use of RP can greatly improve the overall system performance.

Figure 16 and Figure 17 show the results at heavy workload (400 clients). Consistent with the results in Figure 14 and Figure 15,  $R_R$  for RP is much lower than that for EWMA.  $D_R$  for RP is also much lower although they are higher than that in Figure 15 due to the heavier workload. The improvement due to RP is greater at heavy workload as the broadcast selection becomes more critical. Figure 18 to Figure 25 show the performance of RP and EWMA under different workload and drop bounds. As can be observed in the figures,  $R_R$  for RP is consistently smaller than that for EWMA in all the cases.  $D_R$  for RP is smaller than that for EWMA except for the case of medium workload and looser deadlines (Figure 23). In this case, the importance of the broadcast algorithm is less and giving higher priorities to shorter transactions may increase the total drop rate as a result of a larger number of transactions missing their deadlines.

If we compare the results between the access frequencies based algorithms (EWMA & RP) with the deadline-based algorithms (bEDF and bEQSD), it can be observed that  $R_R$  for the deadline-based algorithms are generally better than the access frequencies based algorithms except at looser deadline constraints. It is because the deadline-based algorithms use a more effective policy to schedule the data items for broadcast

and better reflect the urgency of the transactions. The waiting time of the transactions with similar deadline constraints are about the same. However, in the access frequencies based algorithms, the waiting time of the transactions can vary substantially depending on the “hotness” of the requested data items.

We note further that the performance of the deadline-based algorithms is similar to the access frequencies based algorithms in meeting the deadline constraints of the transactions ( $D_R$ ) when the workload is not heavy.  $D_R$  for the deadline-based algorithms are smaller at looser deadline constraints and the workload is medium (Figure 11 and Figure 23). However, it is much poorer at heavy workload and tighter deadline constraints. This is because in the deadline-based algorithms many transactions will miss their deadlines due to the long waiting time for data items at heavy workload. In the access frequencies based algorithms, some of the transactions can be completed before their deadlines if their requesting data items are hot data items even though the workload is very heavy. If the deadline constraints are tighter,  $D_R$  for the deadline-based algorithms are higher than the access frequencies based algorithms also due to insufficient time for getting the data items.

## 6 Conclusions

Research in mobile computing has received growing interest in recent years due to the large number of potential applications. With the support of mobile network, mobile clients can access information from diverse information sources using their notebook, palmtop and personal digital assistant no matter where they are located. Two important issues in the design of mobile computing systems are (1) dissemination of data items to transactions from mobile clients so that the transactions can be completed before their deadlines and (2) provision of consistent data to the transactions. In this paper, we concentrate on the first issue. As for the second issue, we noted that even though many good broadcast algorithms have been suggested by researchers, they do not address the important case of multiple data items per transaction.

In this paper, we examine two common broadcast algorithms for selecting data item to broadcast. The first one is based on access frequencies and the second one is based on transaction deadlines. Based on these two mechanisms, we propose two new broadcast algorithms, called Request Proportion (RP) and Equal Slack Data (EQSD) with the objective to minimize the number of transactions missing their deadlines as well as the mean transaction response time. In our algorithms, the lengths of the transactions are considered explicitly in the selection of data items for broadcast. A detailed mobile computing system model is developed in which the mobile units may have a client cache. Simulation experiments have been

performed to study the performance of our broadcast algorithms, RP and EQSD, as compared with other previously proposed algorithms, the EWMA and batch EDF, under different workload, deadline constraints and broadcast cycle lengths. The results confirmed our belief that in the design of broadcast algorithms, the length of the transactions is a very important factor, and that our broadcast algorithms outperform previously proposed algorithms not designed explicitly to handle multiple data items per request.

## References

- [1] Acharya, S., “Broadcast Disks: Dissemination-based Data Management for Asymmetric Communication Environments”, *Technical Report CS-97-15*, Department of Computer Science, Brown University, Sep. 1997.
- [2] 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.
- [3] 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.
- [4] Franklin, M. and Zdonik, S., “Data In Your Face”: Push Technology in Prospective”, in *Proceedings of 1998 ACM SIGMOD Conference*, Seattle, 1998.
- [5] Hu, Q. and Lee, D.L., “Adaptive Cache Invalidation Methods in Mobile Environments”, in *Proceedings. 6th IEEE International Symposium on High Performance Distributed Computing Environments*, 1997.
- [6] Hameed, S. and Vaidya, N.H., “Efficient Algorithms for Scheduling Single and Multiple Channel Data Broadcast”, Technical Report 97-002, Department of Computer Science, Texas, A&M University, Feb. 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] Imielinski, T., Viswanathan, S. and Badrinath, B.R., “Data on Air: Organization and Access”, *IEEE Transactions on Knowledge and Data Engineering*, vol 9, no. 3, pp. 353-372.
- [9] Leong, H.V. and Si, A., “Database Caching over the Air-Storage”, *The Computer Journal*. Volume 40, number 7, pages 401-415, 1997.
- [10] Lam, K. Y., Au, M.W. & Chan, E., “Broadcast of Consistent Data to Read-Only Transactions from Mobile Clients”, to appear in 2<sup>nd</sup> IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, Feb. 1999.
- [11] Pitoura, E. and Samaras, G., *Data Management for Mobile Computing*, Kluwer Academic Publishers, 1997.
- [12] Stathatos, K., Roussopoulos, N. and Baras, J.S., “Adaptive Data Broadcast in Hybrid Networks”, *Proc. 23<sup>rd</sup> VLDB Conference*, Athens, Greece, 1997.
- [13] Su, C. J., Tassioulas, L., “Broadcast Scheduling for Information Distribution”, *Proc. IEEE Infocom*, 1997, Kobe.
- [14] Ulusoy, O. “Real-Time Data Management for Mobile Computing”, in *Proceedings of International Workshop on Issues and Applications of Database Technology (IADT'98)*, Berlin, Germany, July 1998.

[15] Vaidya,n.H., Hameed, S., "Improved Algorithms for Scheduling Data Broadcast", *Technical Report 96-029*, Department of Computer Science, Texas A& M University.

[16] Wu, K.L., M.S., Yu and Chen, M.S., "Energy-Efficient Caching for Wireless Mobile Computing" in *Proceedings of 12th Intl. Conference on Data Engineering*, 1996.

[17] Xuan, P., O. Gonzalez, J. Fernandez & Ramamritham, K., "Broadcast on Demand: Efficient and Timely Dissemination of Data in Mobile Environments", in *Proceedings of 3<sup>rd</sup> IEEE Real-Time Technology Application Symposium*, 1997.

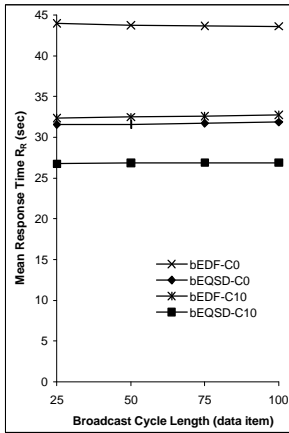


Figure 2. Mean Response Time with Client 200, Drop Period 35~55 sec

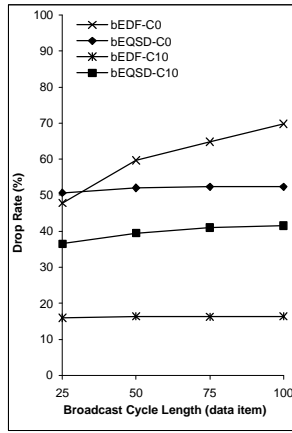


Figure 3. Drop Rate with Client 200, Drop Period 35~55 sec

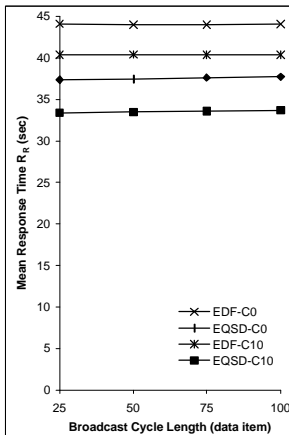


Figure 4. Mean Response Time with Client 400, Drop Period 35~55 sec

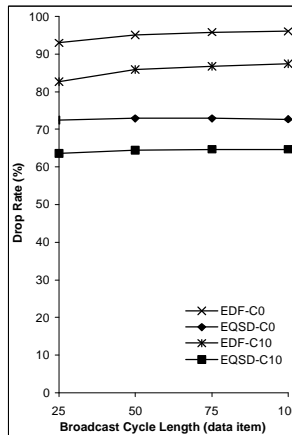


Figure 5. Drop Rate with Client 400, Drop Period 35~55 sec

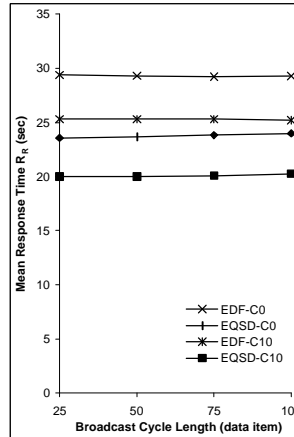


Figure 6. Mean Response Time with Client 200, Drop Period 20~40 sec

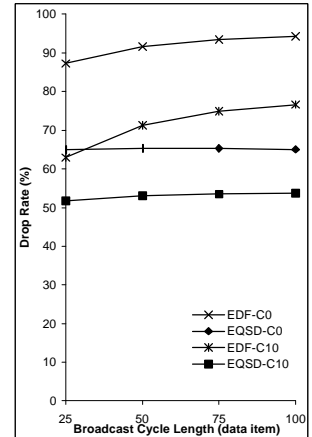


Figure 7. Drop Rate with Client 200, Drop Period 20~40 sec

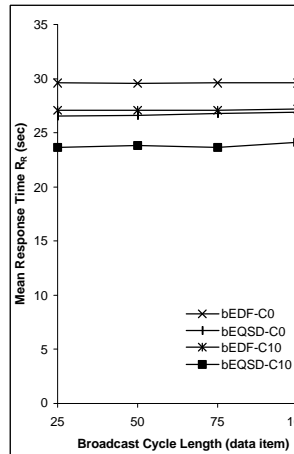


Figure 8 Mean Response Time with Client 400, Drop Period 20~40 sec

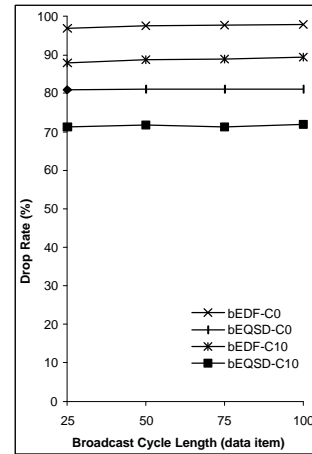


Figure 9 Drop Rate with Client 400, Drop Period 20~40 sec

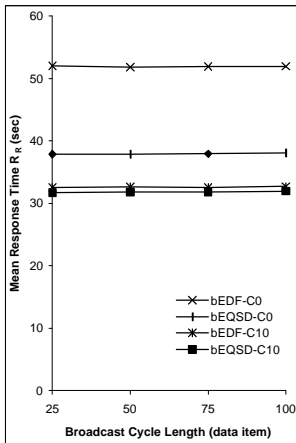


Figure 10. Mean Response Time with Client 200, Drop Period 50~70 sec

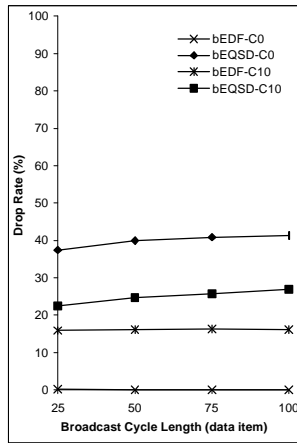


Figure 11. Drop Rate with Client 200, Drop Period 50~70 sec

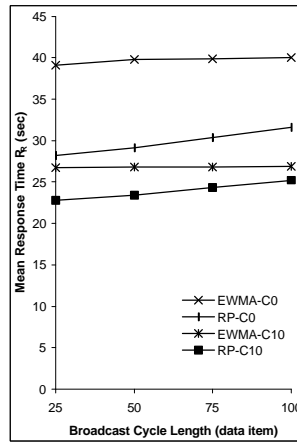


Figure 14. Mean Response Time with Client 200, Drop Period 35~55 sec

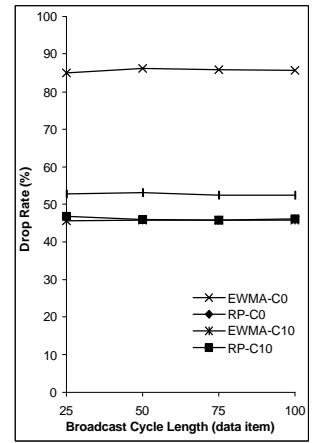


Figure 15. Drop Rate with Client 200, Drop Period 35~55 sec

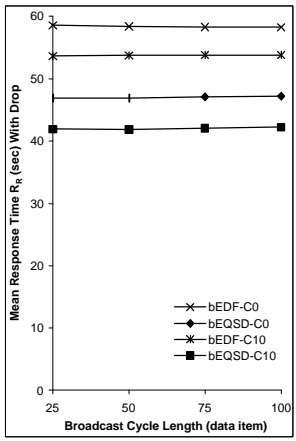


Figure 12. Mean Response Time with Client 400, Drop Period 50~70 sec

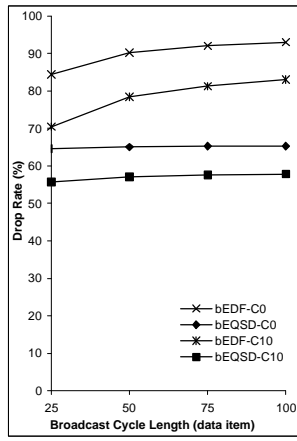


Figure 13. Drop Rate with Client 400, Drop Period 50~70 sec

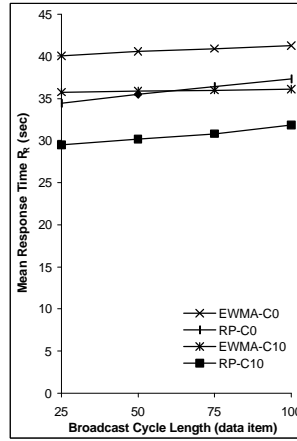


Figure 16. Mean Response Time with Client 400, Drop Period 35~55 sec

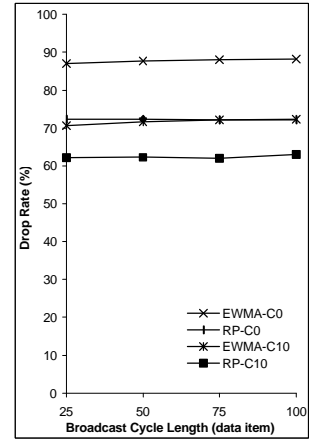


Figure 17. Drop Rate with Client 400, Drop Period 35~55 sec

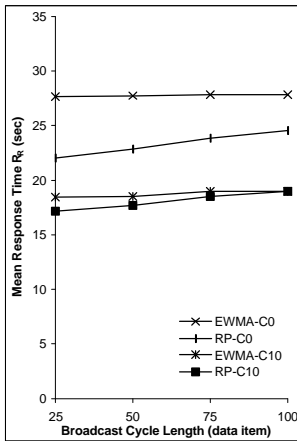


Figure 18. Mean Response Time with Client 200, Drop Period 20~40 sec

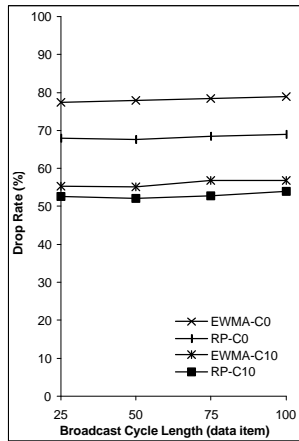


Figure 19. Drop Rate with Client 200, Drop Period 20~40 sec

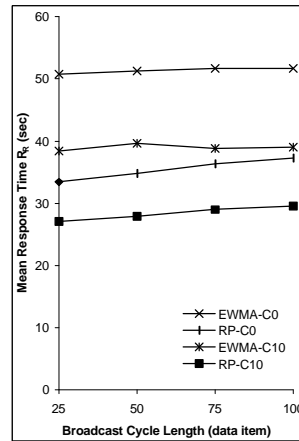


Figure 22. Mean Response Time with Client 200, Drop Period 50~70 sec

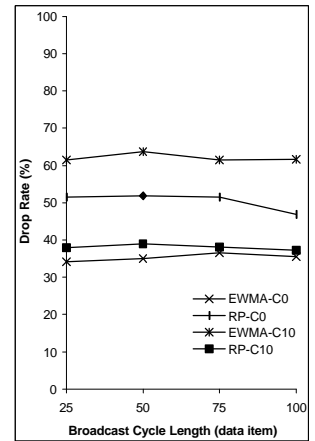


Figure 23. Drop Rate with Client 200, Drop Period 50~70 sec

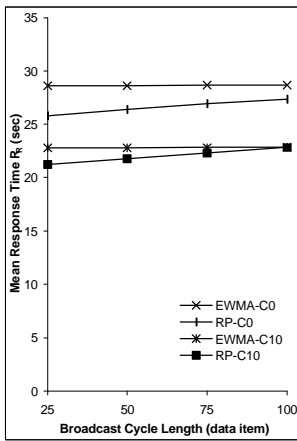


Figure 20. Mean Response Time with Client 400, Drop Period 20~40 sec

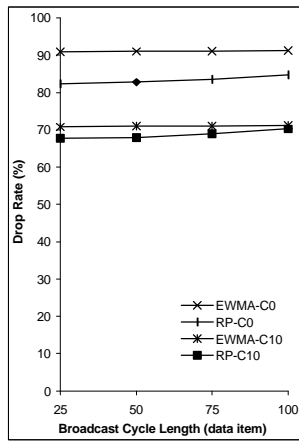


Figure 21. Drop Rate with Client 400, Drop Period 20~40 sec

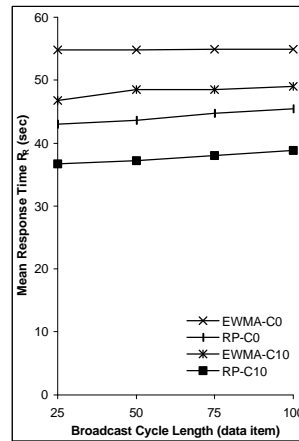


Figure 24. Mean Response Time with Client 400, Drop Period 50~70 sec

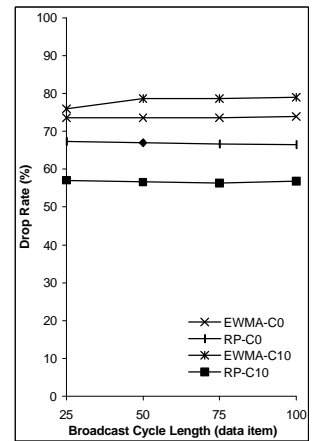


Figure 25. Drop Rate with Client 400, Drop Period 50~70 sec