Data Broadcast in Asymmetric Wireless Environments*

Nitin H. Vaidya Sohail Hameed
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
E-mail: {vaidya,shameed}@cs.tamu.edu
Phone: (409) 845-0512
FAX: (409) 847-8578

Abstract

With the increasing popularity of portable wireless computers, mechanisms to efficiently transmit information to such *clients* are of significant interest. The environment under consideration is *asymmetric* in that the information server has much more bandwidth available, as compared to the clients. In such environments, often it is not possible (or not desirable) for the clients to send explicit requests to the server. It has been proposed that in such systems the server should broadcast the data periodically. One challenge in implementing this solution is to determine the *schedule* for broadcasting the data, such that the wait encountered by the clients is minimized. A *broadcast schedule* determines what is broadcast by the server and when. In this paper, we present algorithms for determining broadcast schedules that minimize the wait time. Simulation results are presented to demonstrate that our algorithms perform well. Variations of our algorithms for environments subject to errors, and systems where different clients may listen to different number of broadcast channels are also considered.

1 Introduction

Mobile computing and wireless networks are fast-growing technologies that are making ubiquitous computing a reality. With the increasing popularity of portable wireless computers, mechanisms to efficiently transmit information to such *clients* are of significant interest [13]. For instance, such mechanisms could be used by a *satellite* or a *base station* to communicate information of common interest to wireless hosts. Approaches for determining what to transmit and when, is the subject of this paper. In the environment under consideration, the *downstream* communication capacity, from server to clients, is relatively much greater than the *upstream* communication capacity, from clients to server. Such environments are, hence, called *asymmetric* communication environments [2]. In an asymmetric environment, *broadcasting* the information is an effective way of making the information available simultaneously to a large number of users. For asymmetric environment, researchers have previously proposed algorithms for designing *broadcast schedules* [4, 6, 7, 8, 9, 10, 11, 12, 17, 18, 19]. Two metrics are used to evaluate these algorithms:

- Access time: This is the amount of time a client has to wait for some information that it needs. It is important to minimize the *access time* so as to decrease the idle time at the client. Several researchers have considered the problem of minimizing the access time [4, 6, 10, 11, 12, 7, 3, 2, 18, 19]

- Tuning time: This is the amount of time a client must *listen* to the broadcast until it receives the information it needs. It is important to minimize the *tuning time*, because the power consumption of a wireless client is higher when it is *listening* to the transmissions, as compared to when it is in a *doze* mode [9, 10, 11, 17].

This paper presents an approach to minimize the *access time*. We consider a database that is divided into *information items* (or *items* for short). Thus, a

*Research reported is supported in part by Texas Advanced Technology Program grant 009741-052-C and National Science Foundation grant MIP-9423735.

broadcast schedule specifies when each item is to be transmitted.

The contributions of this paper are as follows:

- *Square-root rule:* We show that the access time is minimized when the frequency of an item (in the broadcast schedule) is *inversely* proportional to the *square-root* of its *size* and directly proportional to the demand for that item (characterized as *demand probability*). This result is a generalized version of a result presented in [4, 18].

  Impact of *errors* on the scheduling policy is also evaluated. In an asymmetric environment, when a client receives an information item containing errors (due to some environmental disturbance), it is not always possible for the client to request retransmission of the information. In this case, the client must wait for the next transmission of the required item. We evaluate how optimal broadcast frequencies of the items are affected in presence of errors.

  We also consider systems where different clients may listen to different number of broadcast channels, depending on how many they can afford. In such an environment, the schedules on different broadcast channels should be coordinated so as to minimize the access time for most clients.

- For each of the broadcast environments (i.e., with or without errors, and with or without multiple broadcast channels), we determine a theoretical lower bound on the achievable access time. This lower bound is used to determine efficacy of proposed scheduling algorithms.

- We propose a simple "on-line" algorithm, based on the above *square-root* rule for all the environments under consideration. The on-line algorithm can be used by the server to determine which item to broadcast next. On-line algorithms are of significant interest as they are easy to adapt to time-varying demands for the information items. The access time achieved by the on-line algorithms is shown to be very close to the theoretical lower bound. Also, performance of our on-line algorithm is significantly better than that proposed previously [17].

- On-line algorithms use a *decision-making mechanism* to determine which information item is to be broadcast next. For the above on-line algorithm, time complexity of the decision mechanism is linear in the number of information items in the database. This may render the algorithms impractical if number of information items is too large. To alleviate this shortcoming, we present a modified on-line algorithm, that provides the ability to trade *access time* with time complexity.

The rest of the paper is organized as follows. Section 2 introduces some terminology. Section 3 derives the *square-root* rule, and presents two on-line algorithms. The impact of errors is analyzed in Section 4. Section 5 considers an environment where different clients may be listening to different number of channels (depending on what they can afford). Section 6 evaluates the performance of our schemes. Related work is summarized in Section 7. A summary is presented in Section 8.

2 Preliminaries

This section introduces much of the terminology and notations to be used in rest of the paper.

- Database at the server is assumed to be divided into many *information items*. The items are not necessarily of the same size. However, results for systems with identical item sizes can be obtained as a special case of the results presented here.

- The time required to broadcast an item of unit length is referred to as one *time unit*. Hence time required to broadcast an item of length l is l time units. Note that unit of length and time unit may be used interchangeably because of the way they are defined.

- M = total number of information items in the server's database. The items are numbered 1 through M.

- l_i represents length of item i.

- To develop a theoretical foundation for our algorithms, we assume that the broadcast consists of cycle of size N time units. The results presented in the paper also apply to *non-cyclic* schedules (for non-cyclic schedules, effectively, $N \rightarrow \infty$).

- *Instance of an item* : An appearance of an item in the broadcast is referred to as an *instance* of the item.

- *Schedule* : *Schedule* for the broadcast cycle is an order of the items in the cycle.

- *Frequency of an item* : *frequency* f_i of item i is the number of instances of item i in the broadcast cycle. The f_i instances of an item are numbered 1 through f_i. Size of the cycle is, therefore, given by $N = \sum_{i=1}^{M} f_i l_i$, where l_i is the length of item i.

- *Spacing* : The *spacing* between two instances of an item is the time it takes to broadcast information from the beginning of the first instance to the beginning of the second instance. s_{ij} denotes the spacing between j-th instance of item i and the next instance of item i $(1 \leq j \leq f_i)$. Note that, after the f_i-th instance of an item in a transmission of the broadcast cycle, the next instance of the same item is the *first* instance in the next transmission of the broadcast cycle.

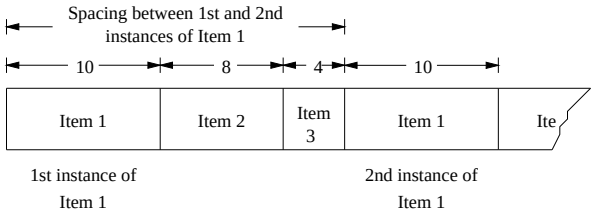


Figure 1: Showing a part of broadcast cycle (Example 1)

Example 1: As an example, refer to Figure 1. The figure shows a part of a broadcast cycle, which contains two instances of *item 1*, and one instance each of *items 2* and *3*. The lengths of the items are 10, 8 and 4 time units respectively. The spacing between the two instances of item 1 is the time from the beginning of first instance of item 1 until the beginning of second instance, which is equal to $10 + 8 + 4 = 22$ time units. Thus, if a client needs item 1 some time (uniformly distributed) between the two instances of item 1, then the average wait is $22/2 = 11$ time units. To reduce this wait, item 1 will have to be transmitted sooner, however, doing so will require one of items 2 or 3 to be transmitted later, causing an increase in the access time of a client needing that item. This example illustrates the need for appropriate scheduling of items in the broadcast. □

- *Item Mean Access Time*: *Item Mean Access Time* of item i, denoted t_i, is defined as the average wait by a client needing item i until it starts receiving item i from the server. Provided that a client is equally likely to need an item i at any instant of time, t_i can be obtained as,

$$t_i = \sum_{j=1}^{f_i} \frac{s_{ij}}{2} \frac{s_{ij}}{N} = \frac{1}{2} \sum_{j=1}^{f_i} \frac{s_{ij}^2}{N},$$

$$\text{where} \qquad N = \sum_{i=1}^{M} f_i l_i$$

If all the f_i instances of item i are equally spaced, that is, for some constant s_i, $s_{ij} = s_i$ $(1 \leq j \leq f_i)$, then, it follows that, $s_i = N/f_i$. In this case, the expression for t_i can be simplified as follows:

$$t_i = \frac{1}{2} \sum_{j=1}^{f_i} \frac{s_{ij}^2}{N} = \frac{1}{2} \sum_{j=1}^{f_i} \frac{s_i^2}{N} = \frac{1}{2} f_i \left(\frac{s_i^2}{N}\right) \frac{1}{2} s_i,$$

$$= \frac{1}{2} s_i, \quad \text{as} \quad s_i = N/f_i \qquad (1)$$

- *Demand probability* : *Demand probability* p_i denotes the probability that an item needed by a client is item i.

- *Overall Mean Access Time*: *Overall Mean Access Time*, denoted t, is defined as the average wait encountered by a client (averaged over all items). Thus,

$$t = \sum_{i=1}^{M} t_i p_i = \sum_{i=1}^{M} \left(\frac{1}{2} \sum_{j=1}^{f_i} \frac{s_{ij}^2}{N} \right) p_i$$

When $s_{ij} = s_i$ $(1 \leq j \leq f_i)$, the above equation reduces to

$$t = \frac{1}{2} \sum_{i=1}^{M} s_i p_i \qquad (2)$$

Note:

All results presented here remain valid if p_i in the above expression is replaced by w_i, where w_i may be interpreted as *weight* of item i. The weights of all items do not have to add to 1 (the fact that the p_i's add to 1 does not have any impact on our algorithms). For instance, weight w_i may be obtained as a product of the "priority" of item i and the demand probability of item i. When p_i is replaced by w_i, t is interpreted as a *cost metric* instead of *access time*.

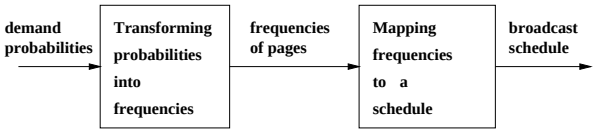


Figure 2: Constructing a Broadcast Schedule

3 Proposed Scheduling Schemes

Figure 2 depicts an abstract view of the procedure for constructing a broadcast schedule. The first block in Figure 2 maps the demand probability distribution into "optimal" item frequencies. Recall that frequency of an item is the number of times the item is to be broadcast in a cycle. Having determined the optimal frequencies, second block in Figure 2 uses the frequencies to determine the broadcast schedule. Our goal is to perform the functions of the two blocks in such a way that *overall mean access time*, t, is minimized. Note that Figure 2 gives a low-level abstraction of the procedure. This helps in obtaining an expression for *optimal* overall mean access time. Algorithms presented here do not use this two-step procedure, however, they are formulated based on results obtained from an analysis of the above procedure.

3.1 Mapping Demand Probabilities to Item Frequencies

We first present theoretical results that motivate our scheduling schemes. The first observation stated in Lemma 1 below is intuitive. This observation also follows from a result presented in [12], and has been implicitly used by others (e.g., [3, 4, 18]).

Lemma 1 *The broadcast schedule with minimum* overall mean access time *results when the instances of each item are equally spaced.*

Proof of the lemma is omitted here for brevity. In reality, it is not always possible to space instances of an item equally. However, the above lemma provides a basis to determine a lower bound on achievable overall mean access time. Note that, while Lemma 1 suggests that spacing between consecutive instances of item i should be constant, say s_i, s_i need not be identical to the spacing s_j between instances of another item j.

The objective in this section is to determine the optimal frequencies (f_i's) as a function of the probability distribution (p_i's) and the length distribution (l_i's). We assume the ideal situation, as implied by Lemma 1, where instances of all items can be equally spaced. This assumption, although often difficult to implement, does lead to a useful result state in Theorem 1. This result is a generalization of a result derived in [4, 18]. The result in [4, 18] applies only to items of identical size, whereas, our result applies to items of differing sizes. We use this result to design *on-line* broadcast scheduling algorithms, which have not been investigated previously.

Theorem 1 Square-root Rule: *Given the demand probability p_i of each item i, the minimum* overall mean access time*, t, is achieved when frequency f_i of each item i is proportional to $\sqrt{p_i}$ and inversely proportional to $\sqrt{l_i}$, assuming that instances of each item are equally spaced. That is,*

$$f_i \propto \sqrt{\frac{p_i}{l_i}}$$

Proof: [15] presents the proof. □

For cycle size N, $\sum_{j=1}^{M} f_j l_j = N$. Therefore, the above theorem implies that, $f_i = \left(N\sqrt{p_i/l_i}\right) / \left(\sum_{j=1}^{M} \sqrt{p_j l_j}\right)$. Also, as spacing $s_i = N/f_i$, a consequence of the above result is that, for *overall mean access time* to be minimized, we need

$$s_i \propto \frac{\sqrt{l_i}}{\sqrt{p_i}}$$

As shown in [15], from Theorem 1 it follows that, the optimal *overall mean access time*, named t_{optimal}, is:

$$t_{\text{optimal}} = \frac{1}{2}\left(\sum_{i=1}^{M} \sqrt{p_i l_i}\right)^2 \qquad (3)$$

t_{optimal} represents a *lower bound* on achievable overall mean access time. As the lower bound is derived by assuming that instances of each item are equally spaced, the bound, in general, is not achievable. However, as shown later, it is possible to achieve performance almost identical to the above lower bound.

Now we present two scheduling algorithms. The first "on-line" algorithm determines which item should be broadcast next by the server. The second on-line algorithm distributes the items into different "buckets", to reduce time complexity of on-line decision-making.

3.2 On-line Scheduling Algorithm

Whenever the server is ready to transmit a new item, it calls the *on-line* algorithm presented here.

The on-line algorithm determines the item to be transmitted next using a *decision rule* – this decision rule is motivated by the result obtained in Theorem 1. As noted previously, Theorem 1 implies that, for optimal performance, instances of an item i should be equally spaced with spacing s_i, where $s_i \propto \sqrt{l_i/p_i}$. This can be rewritten as

$$\frac{s_i^2 p_i}{l_i} = \text{constant}, \quad \forall i, 1 \leq i \leq M \tag{4}$$

The above observation is used in our algorithm, as presented below. We first define some notation. Let Q denote the current time; the algorithm below decides which item to broadcast at time Q. Let $R(j)$ denote the time at which an instance of item j was most recently transmitted; if item j has never been broadcast, $R(j)$ is initialized to -1.[1] Note that, $R(j)$ is updated whenever item j is transmitted. Let function $F(j)$ denote $(Q - R(j))^2 \, p_j/l_j$, $1 \leq j \leq M$. The first on-line algorithm is named Algorithm A.

Algorithm A: ON-LINE algorithm:

Step 1: Determine maximum $F(j)$ over all items j, $1 \leq j \leq M$.
Let F_{max} denote the maximum value of $F(j)$.
Step 2: Choose item i such that $F(i) = F_{max}$.
If this equality holds for more than one item, choose any one of them arbitrarily.
Step 3: Broadcast item i at time Q.
Step 4: $R(i) = Q$.

$Q - R(i)$ is the spacing between the current time, and the time at which item i was previously transmitted. Note that, the function

$$F(i) = (Q - R(i))^2 \, \frac{p_i}{l_i}$$

is similar to the term $s_i^2 \, p_i/l_i$ in Equation 4 above. The motivation behind our algorithm is to attempt to achieve the equality in Equation 4, to the extent possible.

Example 2: Consider a database containing 3 items such that $p_1 = 1/2$, $p_2 = 3/8$, and $p_3 = 1/8$. Assume that items have lengths $l_1 = 1$, $l_2 = 2$ and $l_3 = 4$ time units. Figure 3 shows the items recently broadcast by the server (up to time < 100). The above *on-line* algorithm is called to determine the item to be transmitted at time 100. Thus, $Q = 100$. Also, from Figure 3, observe that $R(1) = 95$, $R(2) = 93$, and $R(3) = 96$. The *on-line* algorithm evaluates function $F(j) = (Q - R(j))^2 p_j/l_j$ for $j = 1, 2, 3$ as 12.5, 147/16 (=9.1875) and 0.5, respectively. As $F(j)$ is the largest for $j = 1$, item 1 is transmitted at time 100. □

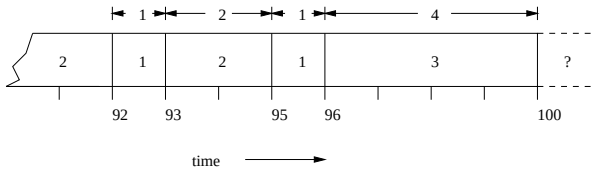


Figure 3: Illustration of the on-line algorithm (Example 2)

It can be shown that, algorithm A always produces a cyclic schedule, if the ties in step 2 of the algorithm are resolved deterministically [15]. Performance measurements for the above algorithm are presented in Section 6. Our algorithm improves access time by a factor of 2 as compared to the *probabilistic* on-line algorithms presented in [17, 18]. In general, the proposed on-line algorithm performs close to the optimal obtained by Equation 3. However, it is also possible to construct scenarios where the schedule produced by the algorithm is not *exactly* optimal, as demonstrated in the next example.

Example 3: Consider the following parameters: $M = 2$, $l_1 = l_2 = 1$, $p_1 = 0.2 + \epsilon$, $p_2 = 1 - p_1$, and $\epsilon < 0.05$. In this case, the on-line algorithm produces the cyclic schedule (1,2), i.e., 1,2,1,2,..., which achieves an overall mean access time of 1.0. On the other hand, the cyclic schedule (1,2,2) achieves overall mean access time $2.9/3 + 2\epsilon/3 < 1$. Thus, in this case, the on-line algorithm is not optimal. However, the overall mean access time 1.0 of the on-line algorithm is within 3.5% of that achieved by the cyclic schedule (1,2,2). □

3.3 On-line Algorithm with Bucketing

A drawback of on-line algorithm A above is the computational cost of *O(M)* required to evaluate F_{max} in step 2 of the algorithm. This cost can be reduced by partitioning the database into "*buckets*" of items, as follows.

Divide the database into k buckets, named B_1 through B_k. Bucket B_i contains m_i items, such that $\sum_{i=1}^{k} m_i = M$, the total number of items in the

[1] The choice of initial value will not affect the mean access time much, unless the broadcast is for a very short time. For broadcasts that last a short time, other initial values may perform better. For instance, $R(j)$ may be initialized to $-\sqrt{l_j}$.

database. We maintain the items in each bucket in a queue. At any time, only items at the front of the buckets are candidates for broadcast at that time. Define $q_j = (\sum_{i \epsilon B_j} p_i)/m_j$ as the average demand probability of the items in bucket B_j, and $d_j = (\sum_{i \epsilon B_j} l_i)/m_j$ as the average length of the items in bucket B_j. Note that $\sum_{i=1}^{k} m_i q_i = 1$. Let Q be the current time and $R(i)$ be the time when item i was most recently broadcast. Let I_j denote the item at the front of bucket B_j. As shown in citevaidya96tech17, for optimality, the following condition must hold when bucketing is used: If item i is in bucket B_j, then

$$\text{spacing } s_i \propto \sqrt{d_j/q_j}$$

In other words,

$$\frac{s_j^2 \, d_j}{q_j} = \text{constant}, \forall j, 1 \leq j \leq M \quad (5)$$

Let $G(j)$ denote $(Q - R(I_j))^2 \, q_j/d_j$, $1 \leq j \leq k$. Function $G(j)$ is analogous to function $F(j)$ used in on-line algorithm A in the previous section. The on-line algorithm with *bucketing*, named Algorithm B, is obtained from the above result.

Algorithm B: ON-LINE WITH BUCKETING:

Step 1: Determine maximum $G(j)$ over all buckets j, $1 \leq j \leq k$. Let G_{max} denote the maximum value of $G(j)$.
Step 2: Choose a bucket B_i such that $G(i) = G_{max}$. If this equality holds for more than one bucket, choose any one bucket arbitrarily.
Step 3: Broadcast item I_i from the front of bucket B_i at time Q.
Step 4: dequeue item I_i at the front of the bucket B_i and enqueue it at the rear of B_i.
Step 5: $R(I_i) = Q$.

The above algorithm is quite similar to the original on-line algorithm A, except that the decision rule is applied only to items at the front of the k buckets. Hence, the algorithm needs to compare values for only k items giving the time complexity of $O(k)$. Observe that all items within the same bucket are broadcast with the same frequency. This suggests that the (p_i/l_i) values of all items in any bucket should be close for good results.

The *Optimal Overall Mean Access Time* resulting from the above algorithm, as shown in [15], is given by

$$t_{\text{opt\_bucket}} = \frac{1}{2} \left(\sum_{j=1}^{k} m_j \sqrt{q_j d_j} \right)^2 \quad (6)$$

Similar to t_{optimal}, $t_{\text{opt\_bucket}}$ is a lower bound on performance achievable with bucketing.

The above equation shows that $t_{\text{opt\_bucket}}$ is dependent upon the selection of values for m_j's under the constraint that $\sum_{j=1}^{k} m_j = M$. Optimizing the bucketing scheme for a given number of buckets k requires that the m_j's be chosen appropriately, such that the above equation is minimized.

For our simulations, we use a heuristic to determine the membership of items to the buckets. The heuristic for determining the membership of an item i to a bucket B_j is as follows:

Let $R_{min} = \min_i \sqrt{p_i/l_i}$ and $R_{max} = \max_i \sqrt{p_i/l_i}$. Let $\delta = R_{max} - R_{min}$. If, for item i, $\sqrt{p_i/l_i} = R_{min}$, then item i is placed in bucket B_1. Any other item i is placed in bucket B_j $(1 \leq j \leq k)$ if $(j-1)\delta/k < (\sqrt{p_i/l_i} - R_{min}) \leq (j\,\delta/k)$. This is pictorially depicted in Figure 4. The above heuristic executes in $O(M)$ time, and needs to be executed once for given probability and length distributions.

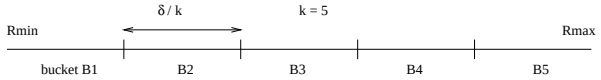


Figure 4: Heuristic for assigning items to k buckets: The interval (R_{min}, R_{max}) is divided into k equal-sized sub-intervals. An item i whose $\sqrt{p_i/l_i}$ value belongs to the j-th sub-interval is assigned to bucket B_j $(1 \leq j \leq k)$.

3.3.1 Comparison of Buckets and Multi-disk [3]

The notion of a *bucket* is similar to that of a *broadcast disk* in the multi-disk approach proposed by Acharya et al. [3]. Therefore, the result in Equation 5 can be used to determine suitable frequencies for the broadcast disks. The differences between the two approaches are as follows: (a) Our algorithm is on-line in that the broadcast schedule is not predetermined. This allows our algorithm to quickly react to any changes in parameters (such as demand probabilities). (b) The algorithm in [3] imposes the constraint that the instances of each item be equally spaced at the risk of introducing *idle* periods (or "holes") in the broadcast schedule (the holes may be filled with other information). Our algorithm also tries to space items at equal spacing, however, it does not enforce the constraint rigidly. Therefore, our algorithm does not create such *holes*. The argument in favor of a

rigid enforcement of equal spacing, as in [3], is that caching algorithms are simplified under such conditions. However, it is possible to implement caching algorithms similar to those in [3] for the bucketing scheme as well. Evaluation of the caching algorithms is beyond the scope of this paper. (c) Our algorithm works well with items of arbitrary sizes. [3] is constrained to fixed size items. (d) Acharya et al. do not have a way of determining the optimal frequencies for the different disks, whereas, our algorithm automatically tries to use the optimal frequencies.

4 Effect of Transmission Errors on Scheduling Strategy

In Section 3, we presented on-line algorithms for determining broadcast schedules. These algorithms do not take into account transmission errors. In this section, we modify our basic approach to design broadcast schedules in the presence of transmission errors.

In the discussion so far, we assumed that each item transmitted by the server is always received correctly by each client. As the wireless medium is subject to disturbances and failures, this assumption is not necessarily valid. Traditionally, in an environment that is subject to failures, the data is encoded using error control codes (ECC). These codes enable the client to "correct" some errors, that is, recover data in spite of the errors. However, ECC cannot correct large number of errors in the data. When such errors are detected (but cannot be corrected by the client), the server is typically requested to retransmit the data.

In the asymmetric environment under consideration here it is not always possible for the client to ask the server to retransmit the data.[2] If a client waiting for item i receives an instance of item i with *uncorrectable* errors, the item is discarded by the client. The client must wait for the next instance of item i. In this section, we evaluate the impact of uncorrectable errors on the scheduling strategy for broadcasts.

Suppose that uncorrectable errors occur in an item of length l with probability $E(l)$.[3] [15] shows that the *overall mean access time*, t, for this case, assuming that instances of item i are equally spaced with spacing s_i, is given by

$$t = \frac{1}{2} \sum_{i=1}^{M} s_i p_i \left(\frac{1 + E(l_i)}{1 - E(l_i)} \right) \quad (7)$$

The *Square Root Rule* in Theorem 1 needs to be modified to take errors into account as follows :

Theorem 2 *Given that the probability of occurrence of uncorrectable errors in an item of length l is $E(l)$, the* overall mean access time *is minimized when*

$$f_i \propto \sqrt{\frac{p_i}{l_i}} \left(\frac{1 + E(l_i)}{1 - E(l_i)} \right)^{1/2}$$

and

$$s_i \propto \sqrt{\frac{l_i}{p_i}} \left(\frac{1 + E(l_i)}{1 - E(l_i)} \right)^{-1/2}$$

Proof : See [15] for proof. □

The lower bound on overall mean access time now becomes,

$$t_{opt\_error} = \frac{1}{2} \left(\sum_{i=1}^{M} \sqrt{p_i l_i} \left(\frac{1 + E(l_i)}{1 - E(l_i)} \right)^{1/2} \right)^2 \quad (8)$$

For the purpose of demonstration, we now consider a simple error model. Let *uncorrectable* errors occur according to a Poisson process with rate λ per unit time. Thus, $E(l_i) = 1 - e^{-\lambda l_i}$. Substituting this expression into Equation 7, and simplifying, yields

$$t = \frac{1}{2} \sum_{i=1}^{M} s_i \, p_i \, \left(2e^{\lambda l_i} - 1 \right) \quad (9)$$

Using Theorem 2, the optimal spacing s_i for an item i should be such that

$$s_i \propto \sqrt{\frac{l_i}{p_i}} \left(2e^{\lambda l_i} - 1 \right)^{-1/2}$$

This implies that

$$\frac{s_i^2 \, p_i}{l_i} \left(2e^{\lambda l_i} - 1 \right) = \text{constant}$$

The on-line scheduling algorithms presented previously can be trivially modified to take into account the above result. For instance, Algorithm A can be used as such with the exception that function $F(j)$ needs to be re-defined as $F(j) = (Q - R(j))^2 \, (p_j / l_j) \, \left(2e^{\lambda l_i} - 1 \right)^{1/2}$, $1 \leq j \leq M$. Section 6 evaluates the modified algorithm A (using the re-defined function $F(j)$).

[2] Even if it were possible for a client to send a retransmit request to the server, it is not clear that a broadcast scheme should allow such requests, because it is possible that many clients receive the original broadcast correctly, but only a few do not due to some localized disturbance.

[3] Now, l_i denotes length of item i after encoding with an error control code.

5 Multiple Broadcast Channels

The discussion so far assumed that the server is broadcasting items over a single channel and all the clients are tuned to this channel. One can also conceive an environment in which the server has a large available bandwidth which is divided into multiple channels, the channels being numbered 1 through c. The clients can then subscribe to as many channels as they want (and can afford). It is apparent that proper use of these channels should result in better performance. Here we give one approach to exploit these channels to improve performance. Other approaches are also possible [15].

The approach considered here uses a modification of Algorithm A, described in Section 3.2, to accommodate multiple channels. Let the total number of broadcast channels be c, the channels being numbered 1 through c. A client capable of listening to, say, n broadcast channels, is assumed to be listening to channels 1 through n.

The scheduling scheme for multiple channels works as follows. Items to be broadcast over channel 1 are determined using on-line algorithm A in Section 3.2, without any modifications to the algorithm. For channel 2, items to be broadcast are also determined using algorithm A, but with a different interpretation of $R(i)$ for item i. Let us define $R_j(i)$ as the instant when item i was last broadcast over channel j. For channel 2, $R(i)$ is defined as the maximum of $R_1(i)$ and $R_2(i)$, $1 \leq i \leq M$. This $R(i)$ is used in algorithm A to determine items to be broadcast on channel 2. Similarly, for channel 3, we define $R(i)$ to be maximum of $R_1(i)$, $R_2(i)$ and $R_3(i)$, and use it in on-line algorithm A.

More formally, the algorithm for determining items to be broadcast on a channel h is as described below. Here, Q is the time at which an item to be broadcast on channel h is to be determined. Function $F(i)$ is defined here as $(Q - R(i))^2 \, p_i/l_i$, where $R(i)$ is defined appropriately for each channel h.

On-line algorithm for channel h, $1 \leq h \leq c$:

Step 1: $R(i) = max_{1 \leq j \leq h}\{R_j(i)\}, \quad 1 \leq i \leq M$.
Step 2: Determine maximum $F(j)$ over all items j, $1 \leq j \leq M$. Let F_{max} denote the maximum value of $F(j)$.
Step 3: Choose item i such that $F(i) = F_{max}$. If this equality holds for more than one item, choose any one of them arbitrarily.
Step 4: Broadcast item i on channel h at time Q.
Step 5: Set $R_h(i) = Q$.

A heuristic to initialize the values of $R_j(i)$, $1 \leq j \leq c$, $1 \leq i \leq M$, is given in the Appendix. Section 6 evaluates this algorithm.

The above algorithm can be easily modified to incorporate *bucketing*, as in algorithm B, to reduce the time complexity. Modifying the maximum taken in step 1 above to $1 \leq j \leq c$ (instead of $1 \leq j \leq h$) yields an algorithm with a different behavior. We are currently generalizing the above algorithm to optimize the weighted access time where weights used are the probability ν_i of a client listening to first i channels [16].

6 Performance Evaluation

In this section, we present simulation results for various algorithms presented above. Each simulation was conducted for at least 8 million item requests by the clients. Other parameters used in the simulation are described below.

6.1 Demand Probability Distribution

We assume that the demand for various items at the client follows Zipf distribution (similar assumptions are made by other researchers as well [1, 2, 3, 4, 18]). The Zipf distribution may be expressed as follows:

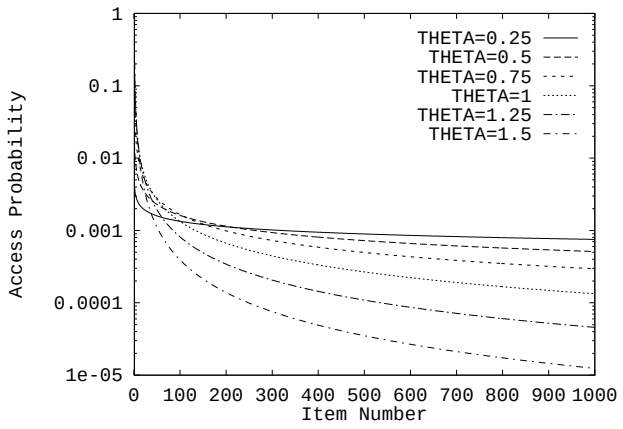
$$p_i = c \left(\frac{1}{i}\right)^{\theta}, \qquad 1 \leq i \leq M$$

where $c = 1/\sum_{i=1}^{M}(1/i)^{\theta}$ is a normalizing factor, and θ is a parameter named *access skew coefficient*. Different values of the access skew coefficient θ yield different Zipf distributions. For $\theta = 0$, the Zipf distribution reduces to uniform distribution with $p_i = 1/M$. However, the distribution becomes increasingly "skewed" as θ increases (that is, for larger θ, the range of p_i values becomes larger). Different Zipf probability distributions resulting from different θ values are shown in Figure 5(a).

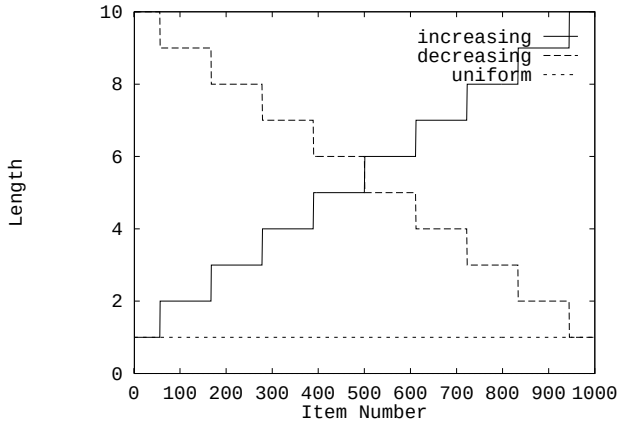
6.2 Length Distribution

A *length distribution* specifies length l_i of item i as a function of i, and some other parameters. In this paper, we consider the following length distribution.

$$l_i = \text{round}\left(\left(\frac{L_1 - L_0}{M - 1}\right)(i - 1) + L_0\right), \qquad 1 \leq i \leq M$$



(a) Zipf Distribution for various values of θ



(b) Length Distribution

Figure 5: (a) shows the Zipf Distribution for various values of access skew coefficient θ. Note that the scale on vertical axis is logarithmic. The probability distribution becomes more skewed with increasing θ. (b) shows three length distributions used in our analysis.

where L_0 and L_1 are parameters that characterize the distribution. L_0 and L_1 are both non-zero integers. `round()` function above returns a rounded integer value of its argument.

We consider three special cases of the above length distribution, obtained by choosing appropriate L_0 and L_1 values.

- *Uniform Length Distribution :* In this case, $L_0 = L_1 = 1$. The distribution reduces to $l_i = 1$, $1 \leq i \leq M$.
- *Increasing Length Distribution :* In this case, $L_0 = 1$ and $L_1 = 10$. In this case, l_i is a non-decreasing function of i, such that $1 \leq l_i \leq 10$, $1 \leq i \leq M$.
- *Decreasing Length Distribution :* In this case, $L_0 = 10$ and $L_1 = 1$. In this case, l_i is a non-increasing function of i, such that $1 \leq l_i \leq 10$, $1 \leq i \leq M$.

Figure 5(b) plots the three length distributions. In addition to these length distributions, we also use a *random* length distribution obtained by choosing lengths randomly distributed from 1 to 10 with uniform probability.

6.3 Request Generation

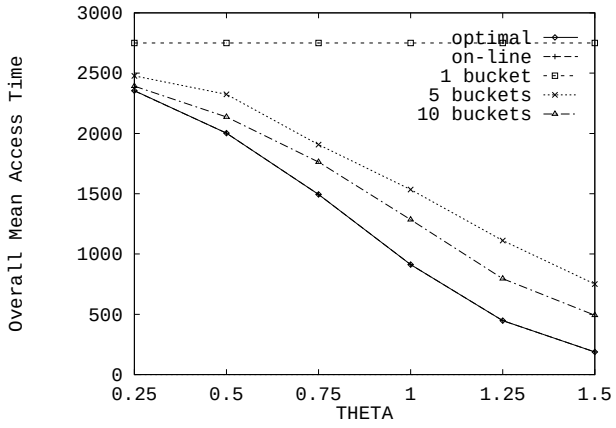
For our simulations, we generated 2 requests for items per time unit. Simulation time is divided into intervals of unit length; 2 requests are generated during each such interval. The time at which the requests are made is uniformly distributed over the corresponding unit length interval. The items for which the requests are made are determined using the demand probability distribution.

6.4 Performance Evaluation in the *Absence* of Uncorrectable Errors

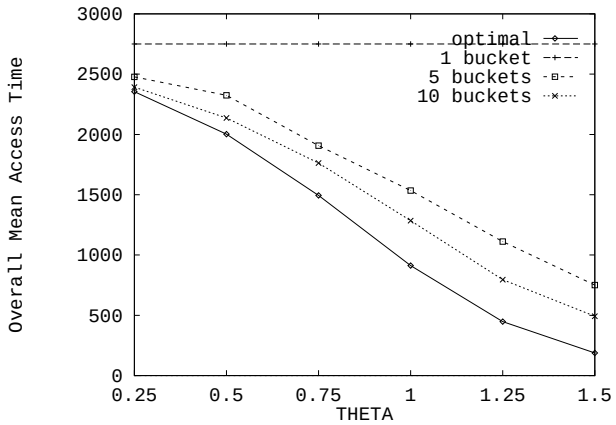
In this section, we evaluate Algorithms A and B, assuming that uncorrectable transmission errors do not occur. Performance evaluation in presence of such errors is discussed in the next section.

Figures 6, 7 and 8 plot *overall mean access time* for different values *of access skew coefficient* θ, for three length distributions presented earlier. (The results for uniform length distribution are similar [15, 16].) In each of these figures, part (a) plots the simulation results and (b) plots the analytical results. In part (a), the curve labeled "Algo A" corresponds to the access time obtained by simulating algorithm A. The curves

With Increasing Length Distribution



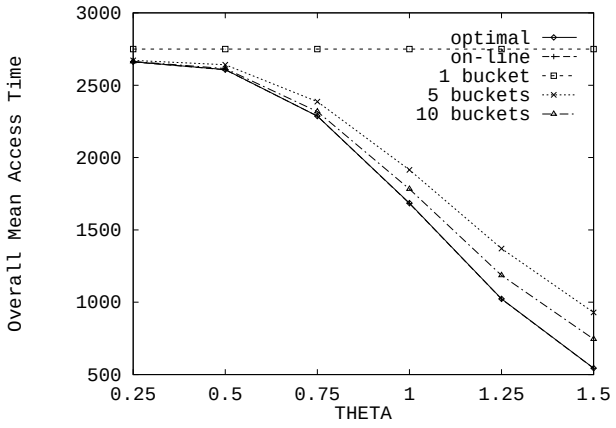
(a) Simulation results



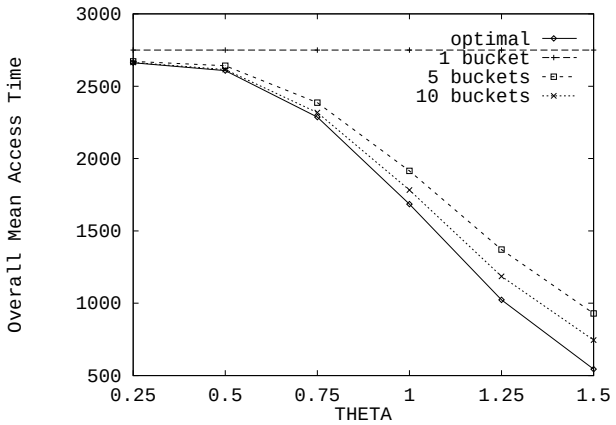
(b) Analytical results

Figure 6: *Overall mean access time* for different values of access skew coefficient θ and using increasing length distribution. In (a), curves for *on-line* and *optimal* overlap with each other. The simulation results are within 0.5% of analytical results.

With Decreasing Length Distribution



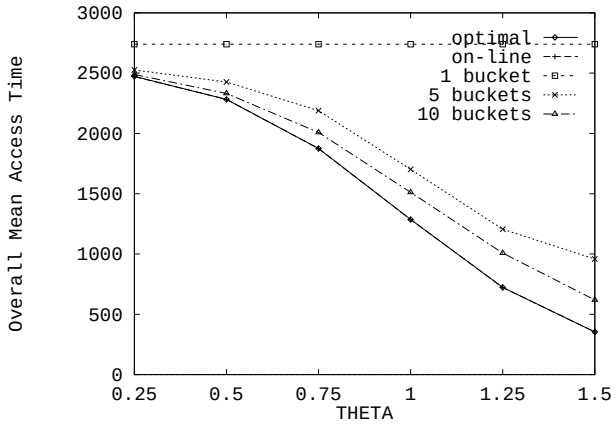
(a) Simulation results



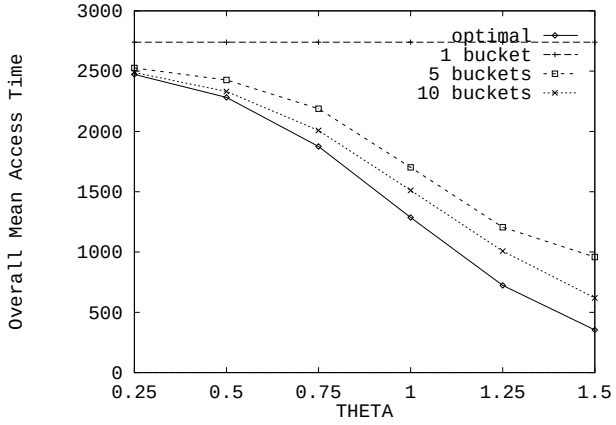
(b) Analytical results

Figure 7: *Overall mean access time* for different values of access skew coefficient θ and using decreasing length distribution. In (a), curves for *on-line* and *optimal* overlap with each other The simulation results are within 0.2% of analytical results..

With Random Length Distribution



(a) Simulation results



(b) Analytical results

Figure 8: *Overall mean access time* for different values of access skew coefficient θ and using random length distribution. In (a), curves for *on-line* and *optimal* overlap with each other. The simulation results are within 0.3% of analytical results.

labeled "i bucket" in part (a) correspond to the access time obtained by simulating algorithm B using i buckets. In part (a) and (b) both, the curve "optimal" corresponds to t_{optimal} obtained using Equation 3. In part (b) of each figure, the curve labeled "i buckets" corresponds to $t_{\text{opt\_bucket}}$ obtained using Equation 6 using i buckets.

First observation, as noted in the caption of each figure, is that the simulation results are very close to the corresponding analytical results. Now note that, when number of buckets is 1, Algorithm B reduces to the so called "flat" cyclic scheduling [3] scheme where each item is broadcast once in a broadcast cycle. As the number of buckets approaches the number of items M, performance of the bucketing algorithm should approach the performance of algorithm A. As algorithm A has a higher time complexity that algorithm B, it is interesting to see how performance of algorithm B improves when the number of buckets is increased. Observe that, the access time with 5 buckets is much smaller than that with just 1 bucket. However, using 5 buckets is not always adequate to achieve access time of algorithm A. Increasing the number of buckets further to, say, 10 further improves the performance of algorithm B. For large θ (i.e., large skew in probability distribution), number of buckets needs to be larger to achieve performance close to optimal. For instance, for $\theta = 0.75$ and 1, using 5 buckets gives the performance much better than that with using 1 bucket. However, further improvement in performance by doubling the number of buckets is not that significant. But for $\theta = 1.5$, even using 10 buckets does not bring the performance close to optimal. However, in this case, it does improve the performance with much bigger factor.

Thus, the choice of the number of buckets is more critical when the skew in probability distribution is large.

An important conclusion from above results is that, performance of algorithm B, with a relatively small number of buckets (10 buckets in our illustration) is quite close to that achieved by algorithm A (effectively, using $M = 1000$ buckets). This implies that algorithm B can significantly reduce time complexity of on-line decision making, with only a marginal degradation in performance. Knowing the best possible access time (the *optimal* curve in the figures) allows a designer to choose an appropriate number of buckets. Secondly, as simulation results are very close to the analytical results, analytical results can be used as a first order approximation of actual performance. A drawback of the related previous work on "multi-

disks" [3] is that they had no analytical method to determine an appropriate number of disks. Therefore, to choose a suitable number of "disks", time-consuming simulations are necessary.

6.5 Performance Evaluation in the Presence of Uncorrectable Errors

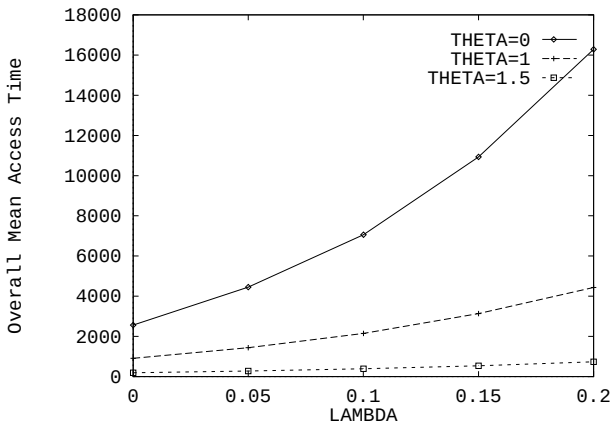
In this section, we evaluate performance of the on-line algorithm in the presence of uncorrectable errors as explained in section 4. Figures 9 and 10 plot *overall mean access time* in the presence of errors for different error rates (λ), and for increasing and decreasing length distributions, respectively. Again, in each of these figures, part (a) plots the simulation results and part (b) plots analytical results for $\theta = 0, 1$ and 1.5. The analytical results are obtained using Equation 8 (substituting $E(l_i) = 1 - e^{-\lambda l_i}$). Note that the results presented in the previous section correspond to the case when λ=0. Also note from theorem 2 that there is no need to take errors into account when lengths of items are uniformly distributed as the factor $(1 + E(l_i))/(1 - E(l_i))$ becomes a constant and hence theorem 2 reduces to theorem 1. Hence the schedule generated is same irrespective of whether errors are considered or not. This is why, we did not bother to simulate for uniform lengths case as it was not worth at all.

From the simulation results, observe that the proposed on-line algorithm A, modified to take errors into account, achieves performance close to optimal. Previous research on broadcasts does not take uncorrectable errors into account when determining the broadcast schedules, or when evaluating the *access time*.

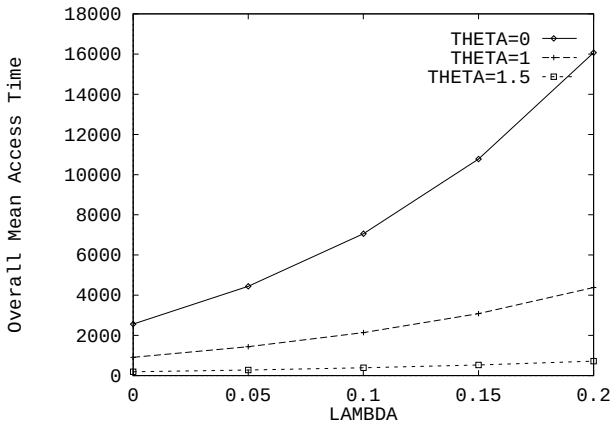
6.6 Performance with Multiple Broadcast Channels

Figure 11 shows the *overall mean access time* against the number of channels for different values of skew coefficient θ and uniform length distribution. The algorithm used to schedule the items on these channels is explained in section 5. In general, as the number of channels increases, the *overall mean access time* decreases, improving the performance of the system. However, the improvement is more significant when there is less skew in access probability, denoted by smaller value of θ. Hence, for $\theta = 0$ the improvement is most significant. Note that for $\theta = 0$, $p_i = 1/M$ = constant, $\forall i$, and since uniform length distribution is used in this simulation, the ratio p_i/l_i and hence spacing s_i is also constant $\forall i$, the resulting

With Increasing Length Distribution

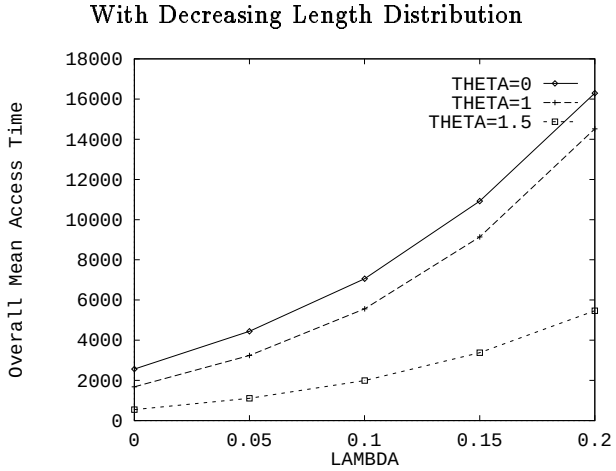


(a) Simulation results

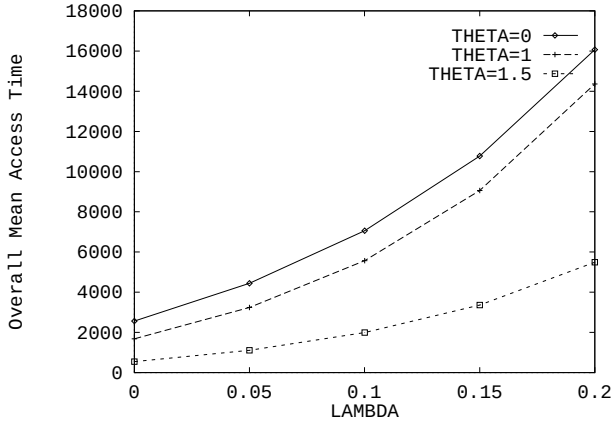


(b) Analytical results

Figure 9: *Overall mean access time* against λ for different values of θ and increasing length distribution. The simulation curves are obtained using Algorithm A modified to take errors into account. The simulation results are within 2.5 % of analytical results.



(a) Simulation results



(b) Analytical results

Figure 10: *Overall mean access time* against λ for different values of θ and decreasing length distribution. The simulation curves are obtained using Algorithm A modified to take errors into account. The simulation results are within 1.1 % of analytical results.

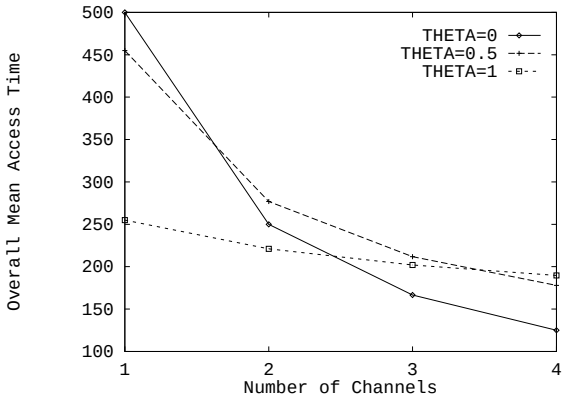


Figure 11: *Overall mean access time* against number of channels for different values of skew coefficient θ and uniform length distribution. For $\theta = 0$, uniform probability of access, the best improvement in performance occurs, whereas for $\theta = 1$, larger skew in access probability, the improvement is not much significant.

broadcast is flat, giving the inverse proportional relationship between *overall mean access time* and the number of channels. However, for higher values of θ, these results do not hold and hence the improvement in *overall mean access time* tends to decrease.

The improvement by increasing the number of channels also depends upon the heuristic used to initialize the $R_j(i)$ values used in *on-line* algorithm for multiple channels. Here, we used the heuristic explained in section 5. The procedure of initializing these values to optimize the *overall mean access time* is an interesting problem and is one of the topics of our on-going research.

7 Related Work

The problem of broadcasting data efficiently has received much attention lately. The existing schemes can be roughly divided into two categories (some schemes may belong to both categories, we have listed them in the most appropriate category): Schemes attempting to reduce the *access time* [4, 3, 2, 1, 8, 12, 7, 6, 18, 19] and schemes attempting to reduce the *tuning time* [10, 9, 11]. However, proposed on-line algorithms have not been studied previously. Also, impact of errors on scheduling, and broadcast on multiple channels, have not been addressed.

Ammar and Wong [4, 18] have performed extensive research on broadcast scheduling and obtained many

interesting results. Our square root rule is a generalization of that obtained by Ammar and Wong. Wong [18] and Imielinski and Viswanathan [8, 17] present an on-line scheme that uses a *probabilistic* approach for deciding which item to transmit. Our on-line algorithm results in an improvement by a factor of 2 in the mean access time as compared to the probabilistic on-line algorithm in [8, 17, 18]. Chiueh [6] and Acharya et al. [3, 2, 1] present schemes that transmit the more frequently used items more often. However, they do not use optimal degree of replication. Our schemes, on the other hand, tend to use optimal frequencies.

Jain and Werth [12] note that reducing the variance of spacing between consecutive instances of an item reduces the mean access time. The two schemes presented in this paper do attempt to achieve a low variance. Similar to our discussion in Section 4, Jain and Werth [12] also note that errors may occur in transmission of data. Their solution to this problem is to use error control codes (ECC) for forward error correction, and a RAID-like approach (dubbed airRAID) that stripes the data. The server is required to transmit the stripes on different frequencies, much like the RAID approach spreads stripes of data on different disks [5]. ECC is not always sufficient to achieve forward error correction, therefore, uncorrectable errors remains an issue (which is ignored in the past work on data broadcast).

8 Summary

This paper considers *asymmetric* environments wherein a server has a much larger communication bandwidth available as compared to the clients. In such an environment, an effective way for the server to communicate information to the clients is to broadcast the information periodically. Contributions of this paper are as follows:

- We proved *square-root rules* (Theorems 1 and 2) which provide a theoretical basis for the proposed algorithms.
- We proposed on-line algorithms for scheduling broadcasts, with the goal of minimizing the *access time*. Simulation results show that our algorithms perform quite well (very close to the theoretical optimal). The *bucketing* scheme proposed in the paper facilitates a trade-off between time complexity and performance of the on-line algorithm.
- The paper considers the impact of errors on optimal broadcast schedules.
- When different clients are capable of listening on different number of broadcast channels, the schedules on different broadcast channels should be designed so as to minimize the access time for all clients. This paper presents an algorithm for scheduling broadcasts in such a system.

More work is needed on some problems discussed in this paper. Future work will also include design of strategies for caching and updates that attempt to achieve optimal performance while incurring low overhead. Further results will be made available at our web site `http://www.cs.tamu.edu/faculty/vaidya/mobile.html`.

Appendix: A heuristic for initializing $R_j(i)$ values

In algorithm A, we assumed that $R(i)$ is initialized to -1 for all i. With only a single broadcast channel, the initial values do not have a significant impact on average access time, particularly when broadcast is performed for extended periods of time. With multiple channels, the initial values assigned to $R_j(i)$ $(1 \leq j \leq c,\ 1 \leq i \leq M)$ play a critical role in determining the access time achieved by using multiple channels. Therefore, appropriate heuristics must be developed to initialize $R_j(i)$ value s. We now describe a simple heuristic to be used while evaluating performance of the above algorithm hereunder :

We first initialize $R_1(i)$, $1 \leq i \leq M$, and then rotate the values by using a staggering factor τ for each subsequent channel, such that $\tau = 1/(c\sum_{j=1}^{M} \frac{p_i}{s_i})$, where p_i is the access probability and $s_i = \sum_{j=1}^{M} \sqrt{p_j l_j}\sqrt{\frac{l_i}{p_i}}$.

The algorithm successively initializes the values of $R_1(i)$, each item one by one. Let us assume that each item is sorted in descending order of $\sqrt{p_i/l_i}$ values.[4] The algorithm is shown below :

Step 1: Set *time*=1
Step 2: For every item $1 \leq i \leq M$ in database
{
Step 3: For every item $1 \leq j < i$
{
Step 4: if $\psi_j \leq 0$
{

[4] if items are not sorted then sort them and renumber the items from 1 through M. The is just a logical renumbering and has no influence over *overall mean access time* measurements whatsoever.

Step 5: $\psi_j = s_j$
Step 6: $time = time + l_j$
}
}
Step 7: $\psi_i = s_i$
Step 8: $R_1(i) = time$
Step 9: $time = time + l_i$
Step 10: For $1 \leq j \leq i$
$\psi_j = \psi_j - l_i$
}
Step 11: Find $R_j(i),\ 1 < j \leq c,\ q \leq i \leq M$, by rotating the values of $R_{j-1}(i)$ by an amount $\tau = 1/(c \sum_{i=1}^{M} \frac{p_i}{s_i})$.

Acknowledgements

We thank referees for their insightful comments.

References

[1] S. Acharya, M. Franklin, and S. Zdonik, "Prefetching from a broadcast disk," in *12th Int. Conf. on Data Engineering*, Feb. 1996.

[2] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks - data management for asymmetric communications environment," in *ACM SIGMOD Conference*, May 1995.

[3] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-based data delivery using broadcast disks," *IEEE Personal Communication*, pp. 50–60, Dec. 1995.

[4] M. H. Ammar and J. W. Wong, "On the Optimality of Cyclic Transmission in Teletex Systems", in *IEEE Transactions on Communications*, Vol. COM-35 No. 1, pp. 68-73, Jan. 1987.

[5] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-performance, reliable secondary storage," *ACM Computing Surveys*, vol. 26, no. 2, pp. 145–185, 1994.

[6] T. Chiueh, "Scheduling for broadcast-based file systems," in *MOBIDATA Workshop*, Nov. 1994.

[7] V. A. Gondhalekar, "Scheduling periodic wireless data broadcast," Dec. 1995. M.S. Thesis, The University of Texas at Austin.

[8] T. Imielinski and S. Viswanathan, "Adaptive wireless information systems," in *Proceedings of SIGDBS (Special Interest Group in DataBase Systems) Conference*, Oct. 1994.

[9] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Power efficient filtering of data on air," in *4th International Conference on Extending Database Technology*, Mar. 1984.

[10] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Energy efficient indexing on air," in ACM SIGMOD, May 1994.

[11] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Data on the air - organization and access," manuscript, 1994.

[12] R. Jain and J. Werth, "Airdisks and airRAID : Modelling and scheduling periodic wireless data broadcast (extended abstract)," Tech. Rep. DIMACS 95-11, Rutgers University, May 1995.

[13] R. J. Douglas (Program Manager), "Battlefield awareness and data dissemination (BADD) program," 1996-2000. Web site at http://maco.dc.isx.com/iso/battle/badd.html.

[14] N. H. Vaidya and S. Hameed, "Data Broadcast Scheduling (Part I)," Tech. Report 96-012, Comp. Sc. Dept., Texas A&M Univ., May 1996.

[15] N. H. Vaidya and S. Hameed, "Data broadcast scheduling: On-line and off-line algorithms," Tech. Rep. 96-017, Computer Sc. Dept., Texas A&M University, College Station, July 1996.

[16] N. H. Vaidya and S. Hameed, Technical Report under preparation.

[17] S. Viswanathan, *Publishing in Wireless and Wireline Environments*. PhD thesis, Rutgers, Nov. 1994.

[18] J. W. Wong, "Broadcast Delivery", in *Proceedings of IEEE*, pp. 1566-1577, Dec. 1988,

[19] Z. Zdonik, R. Alonso, M. Franklin, and S. Acharya, "Are disks in the air, just pie in the sky?," in *IEEE Workshop on Mobile Comp. System*, Dec. 1994.