Scheduling Data Broadcast in Asymmetric Communication 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

Technical Report 96-022

November 1, 1996‡

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 report, 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.

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

†Appears in part at Workshop on Satellite Based Information Services (WOSBIS), Nov. 1996, Rye, NY.

‡Typographical errors in Appendix B corrected on November 18, 1996.

Contents

This technical report is substantially identical to technical report 96-017 [15]. This report presents a new algorithm for multiple channel broadcast, and also presents new performance evaluation results for some algorithms presented in [15]. [15] also presents some multiple channel broadcast algorithms that are not considered in this report.

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. 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 report presents an approach to minimize the *access time*. We consider a database that is divided into *information items* (or *items* for short). Thus, a broadcast schedule specifies when each item is to be transmitted.

The contributions of this report 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 schedule is 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 average 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 each environment under consideration. An 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].

The rest of the report 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 discussed 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 report.

- Database at the server is assumed to be divided into many *information items*. The items are not necessarily of the same size.
- 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 a cycle of size N time units. The results presented in the report 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.

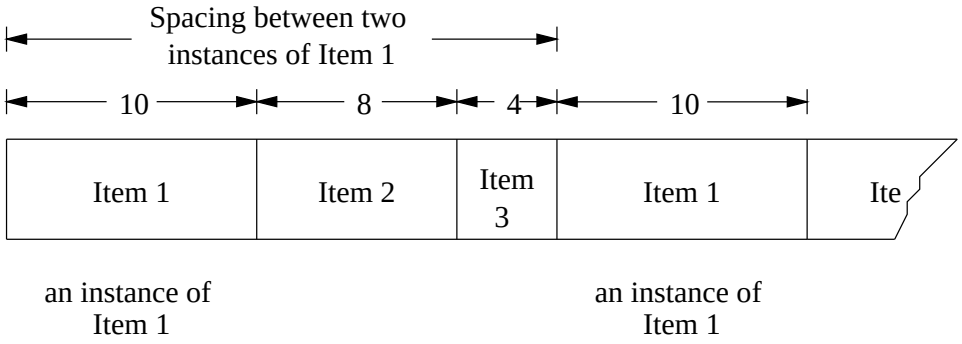


Figure 1: Showing a part of broadcast cycle (Example 1)

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 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 at 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,

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

  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:

$$t_i \;=\; \frac{1}{2}\sum_{j=1}^{f_i} \frac{s_{ij}^2}{N} \;=\; \frac{1}{2}\sum_{j=1}^{f_i} \frac{s_i^2}{N} \;=\; \frac{1}{2}\, f_i \,\left(\frac{s_i^2}{N}\right) \;=\; \frac{1}{2}s_i, \quad \text{as } s_i = N/f_i \qquad (1)$$

- *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

$$t = \frac{1}{2} \sum_{i=1}^{M} s_i \, p_i \tag{2}$$

3 Proposed Scheduling Schemes

3.1 Abstract Procedure

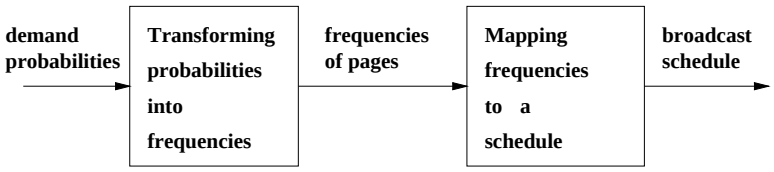


Figure 2: Constructing a Broadcast Schedule

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 broadcast 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 in this report do not use this two-step procedure, however, they are formulated based on results obtained from an analysis of the above procedure.

3.2 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 now 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 stated 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: Appendix A presents the proof. □

Now note that, cycle size $N = \sum_{j=1}^{M} f_j l_j$. 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

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

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

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

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.

3.3 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 $G(j)$ be defined as

$$G(j) = (Q - R(j))^2\, p_j/l_j, \qquad 1 \leq j \leq M$$

The first on-line algorithm is named Algorithm A.

Algorithm A: On-line algorithm:

Step 1: Determine maximum $G(j)$ over all items j, $1 \leq j \leq M$.
Let G_{max} denote the maximum value of $G(j)$.
Step 2: Choose item i such that $G(i) = G_{max}$. 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 $G(i) = (Q - R(i))^2\, p_i/l_i$ is similar to the term $s_i^2\, p_i/l_i$ in Equation 4. 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 $G(j) = (Q - R(j))^2 p_j/l_j$ for $j = 1, 2, 3$ as 12.5, 147/16 (=9.1875) and 0.5, respectively. As $G(j)$ is the largest for $j = 1$, item 1 is transmitted at time 100. □

[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 $-\sqrt{l_j}$.

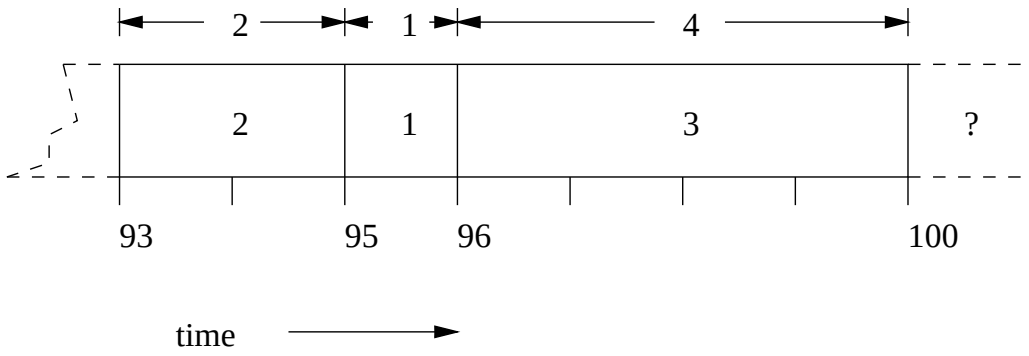


Figure 3: Illustration of the on-line algorithm (Example 2)

It can be shown that, algorithm A 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, as shown in section 6, 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 $0 < \epsilon < 0.05$. 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). □

3.4 On-line Algorithm with Bucketing

A drawback of on-line algorithm A above is the computational cost of *O(M)* required to evaluate G_{max} in step 1 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 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 Appendix B, for optimality, the following condition must hold when bucketing is used: If item i is in bucket B_j, then

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

$$\text{In other words,} \quad \frac{s_i^2\, q_j}{d_j} \quad = \quad \text{constant,} \quad \forall j, \quad 1 \le j \le k \quad \text{and} \quad i \in B_j \tag{5}$$

Let $G(j)$ now denote $(Q - R(I_j))^2\, q_j/d_j$, $1 \le j \le k$. Function $G(j)$ used here is similar (but not identical) to function $G(j)$ used in 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, $1 \le j \le k$.
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 from 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 (in steps 1 and 2) is applied only to items at the front of the k buckets. Hence, the algorithm needs to compare values for only k items resulting in 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 Appendix B, is given by

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

Similar to t_{optimal}, $t_{\text{opt\_bucket}}$ is a lower bound on average access time 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 A_{min} and A_{max} denote the minimum and maximum value of $\sqrt{p_i/l_i}$ $(1 \le i \le M)$, respectively. Let $\delta = A_{max} - A_{min}$. If, for item i, $\sqrt{p_i/l_i} = A_{min}$, then item i is placed in bucket B_1. Any other item i is placed in bucket B_j $(1 \le j \le k)$ if $(j-1)\delta/k \;<\; (\sqrt{p_i/l_i} - A_{min}) \le (j\,\delta/k)$. 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.

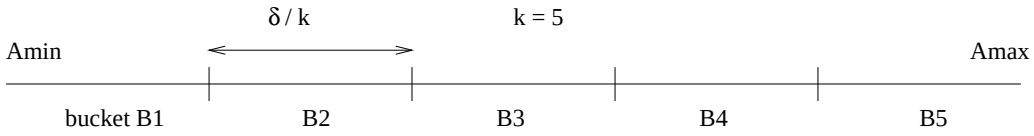


Figure 4: Heuristic for assigning items to k buckets: The interval (A_{min}, A_{max}) is divided into k equal-sized sub-intervals. An item i whose $\sqrt{p_i/l_i}$ value belongs to the j-th sub-interval is assigned to bucket B_j $(1 \leq j \leq k)$.

3.4.1 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.[2] The differences between the two approaches are as follows: (a) Acharya et al. [3] do not have a way of determining the optimal frequencies for the different disks, whereas, our algorithm automatically tries to use the optimal frequencies. (b) 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). (c) 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 report. (d) Our algorithm works well with items of arbitrary sizes. [3] is constrained to fixed size items.

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.[3] 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

[2] As s_i, $i \in B_j$, is proportional to $\sqrt{d_j/q_j}$, it follows that $f_i \propto \sqrt{q_j/d_j}$.

[3] Even if it were possible for a client to send a retransmit request to the server, it is not clear that a broadcast scheme

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 $E(l)$ (Now, l_i denotes length of item i after encoding with an error control code). Appendix C 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

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

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 Appendix C. □

The lower bound on overall mean access time now becomes,

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

Theorem 2 implies that in an optimal schedule,

$$\frac{s_i^2 \, p_i}{l_i} \left(\frac{1 + E(l_i)}{1 - E(l_i)} \right) = \text{constant} \quad , 1 \leq i \leq M$$

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 $G(j)$ needs to be re-defined as $G(j) = (Q - R(j))^2 \; (p_j / l_j) \left(\frac{1+E(l_j)}{1-E(l_j)} \right)$, $1 \leq j \leq M$. Section 6 evaluates the modified algorithm A (using the re-defined function $G(j)$).

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).

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 3.3, 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, may be listening to any n channels.

Let $H = \{1, 2, \cdots, c\}$ denote the set of all broadcast channels. A client (that is interested in the broadcast data) may listen to any non-empty subset S of the set H. For instance, if $c = 2$, then $H = \{1, 2\}$, and S may be $\{1\}$, or $\{2\}$, or $\{1, 2\}$.

Let Π_S denote the probability that S is the set of channels listened to by a client, where $S \subseteq H$. By definition, $\Pi_{\{\}} = 0$ – that is, each client of interest in this discussion listens to at least one channel.

As different clients may be listening to different number of channels, we re-define *overall mean access time* to be an average over all clients. The *overall mean access time* for multichannel broadcast is named $t_{multichan}$, and obtained as,

$$t_{multichan} = \sum_{S \subseteq H} \Pi_S \, t_S$$

where t_S denotes the average access time encountered by a client listening to channels in set S. For instance, when number of broadcast channels is $c = 2$, $H = \{1, 2\}$, and

$$t_{multichan} = \Pi_{\{1\}} \, t_{\{1\}} + \Pi_{\{2\}} \, t_{\{2\}} + \Pi_{\{1,2\}} \, t_{\{1,2\}} \tag{9}$$

Equation 3 presented a lower bound ($t_{optimal}$) on the overall mean access time when a client listens to only 1 channel. Clearly, for a non-empty set of channels S, a lower bound on t_S is given by

$$\frac{t_{optimal}}{|S|}$$

where $|S|$ is the number of channels in set S. It follows that, a lower bound on $t_{multichan}$ is given by

$$t_{multichan\_optimal} = \sum_{S \subseteq H, |S|>0} \Pi_S \, \frac{t_{optimal}}{|S|} \tag{10}$$

In particular, if number of channels $c = 2$, then

$$t_{multichan\_optimal} = \left(\Pi_{\{1\}} + \Pi_{\{2\}} + \frac{\Pi_{\{1,2\}}}{2} \right) t_{optimal}$$

Now we present an on-line algorithm to schedule broadcast on multiple channels. This algorithm is obtained by generalizing algorithm A in Section 3.3. In the following, assume that current time is Q, and the on-line algorithm needs to determine which page to broadcast on channel h (where $1 \leq h \leq c$). Let $R_h(j)$ denote the most recent time when item j was broadcast on channel h ($1 \leq h \leq c$, $1 \leq j \leq M$). $R_h(j)$ is initialized to -1. For a subset S of H, define $R^S(j) = \max_{h \in S} R_h(j)$. Thus, $R^S(j)$ is the time when item j was most recently transmitted on any channel in set S. Similar to the cost function $G(j)$ used in Algorithm A, here we use a function $G_h(j)$ for each channel h ($1 \leq j \leq M$). $G_h(j)$ is defined as follows.

$$G_h(j) = \frac{p_j}{l_j} \left(\sum_{S \subseteq H, h \in S} \Pi_S \left(Q - R^S(j)\right)^2 \right) \tag{11}$$

Note that the summation in the above expression for $G_h(j)$ is over all subsets S of H that contain channel h.

As an illustration, when $c = 2$, we have $H = \{1, 2\}$, and

$$G_1(j) = \frac{p_j}{l_j} \left(\Pi_{\{1\}} (Q - R^{\{1\}}(j))^2 + \Pi_{\{1,2\}} (Q - R^{\{1,2\}}(j))^2 \right)$$

and

$$G_2(j) = \frac{p_j}{l_j} \left(\Pi_{\{2\}} (Q - R^{\{2\}}(j))^2 + \Pi_{\{1,2\}} (Q - R^{\{1,2\}}(j))^2 \right)$$

The proposed on-line algorithm is as follows.

On-line algorithm for <u>channel h</u>, $1 \leq h \leq c$:

Step 1: $R^S(j) = \max_{h \in S} R_h(j), \forall S, \forall j$, $S \subseteq H$, $1 \leq j \leq M$.
Step 2: Determine maximum $G_h(j)$ over all items j, $1 \leq j \leq M$.
Let G_{max} denote the maximum value of $G_h(j)$ over all j.
Step 3: Choose item i such that $G_h(i) = G_{max}$. 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: $R_h(i) = Q$.

Section 6.6 evaluates the performance of the above algorithm for two channels ($c = 2$). Time complexity of steps 1 and 2 above can be reduced by using techniques similar to *bucketing* (in Section 3.4).

6 Performance Evaluation

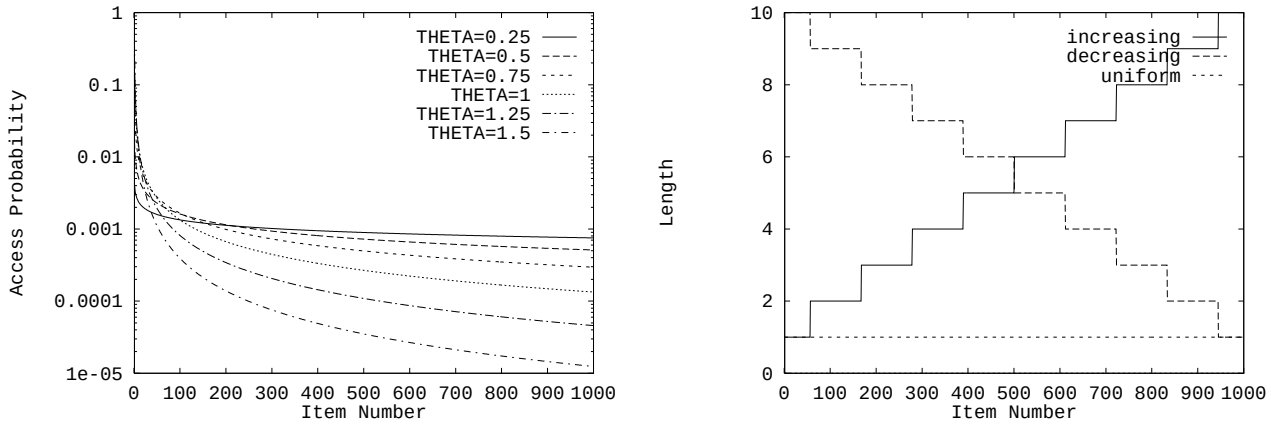
In this section, we present simulation results for various algorithms presented above. Each simulation was conducted for at least 6 million item requests by the clients. Other parameters used in the simulation are described below.

6.1 Demand Probability Distribution

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

$$p_i = \frac{(1/i)^\theta}{\sum_{i=1}^{M}(1/i)^\theta} \qquad 1 \le i \le M$$

where 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).



(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. Another length distribution used in this report is illustrated in Figure 6.

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 report, we consider the following length distribution.

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

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 integral 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. The *random length distribution* used here is shown in Figure 6.

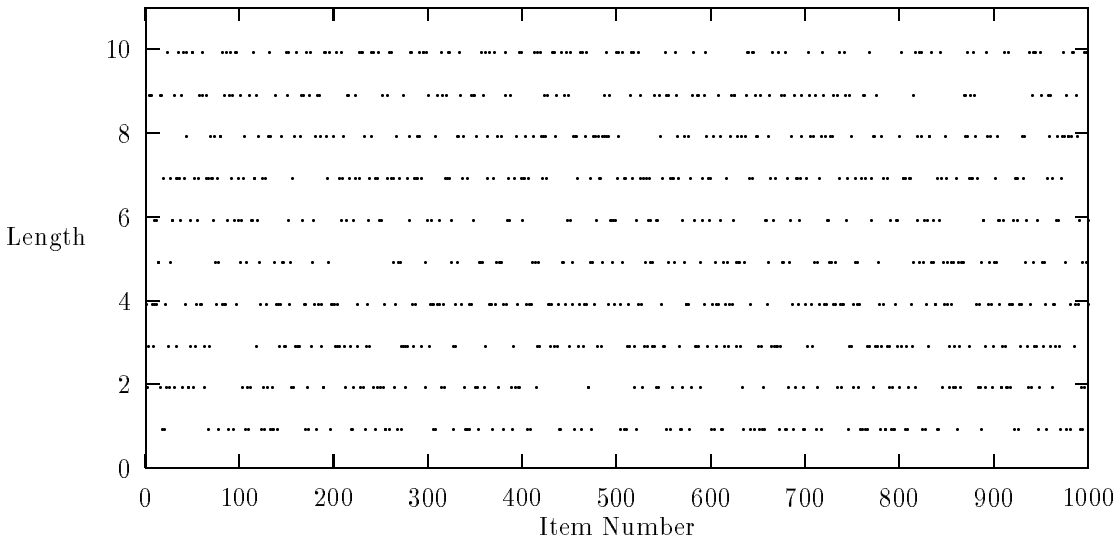


Figure 6: *Random* Length Distribution. Lengths are randomly distributed from 1 to 10 with uniform probability.

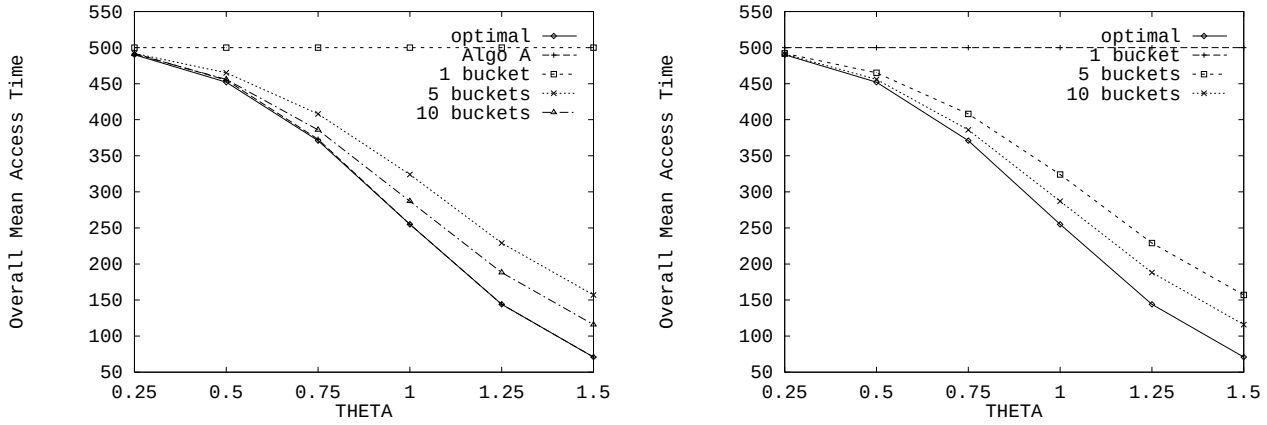
6.3 Request Generation

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.

With Uniform Length Distribution

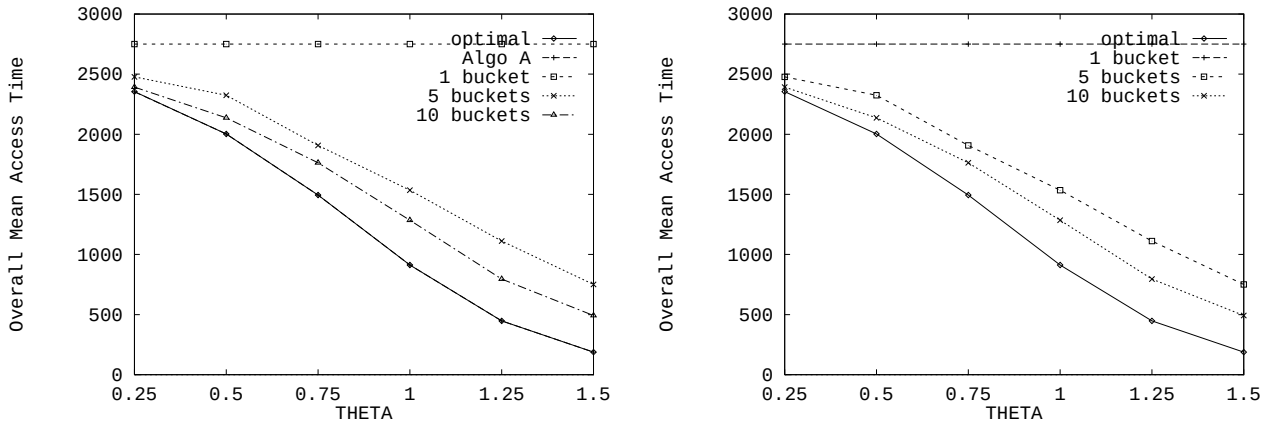


(a) Simulation results

(b) Analytical lower bounds

Figure 7: *Overall mean access time* for different values of access skew coefficient θ and using uniform length distribution. In (a), curves for *Algo A* and *optimal* overlap with each other. The simulation results are within 0.7% of analytical results.

With Increasing Length Distribution

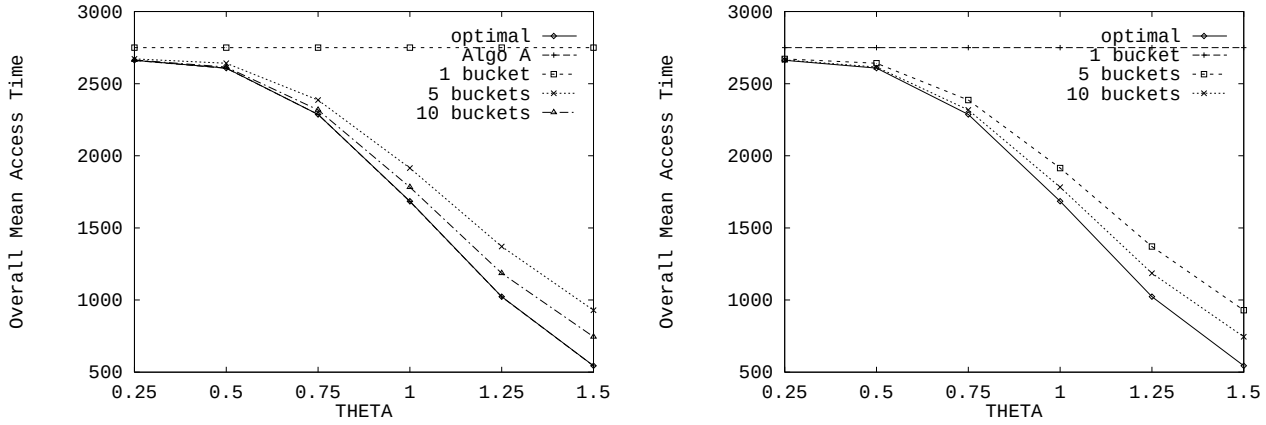


(a) Simulation results

(b) Analytical lower bounds

Figure 8: *Overall mean access time* for different values of access skew coefficient θ and using increasing length distribution. In (a), curves for *Algo A* and *optimal* overlap with each other. The simulation results are within 0.5% of analytical results.

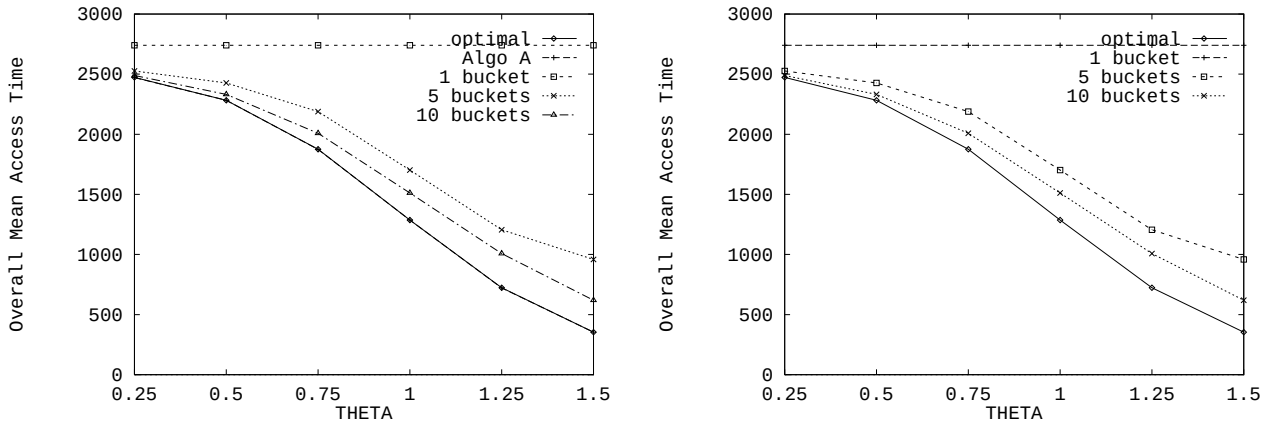
With Decreasing Length Distribution



(a) Simulation results (b) Analytical lower bounds

Figure 9: *Overall mean access time* for different values of access skew coefficient θ and using decreasing length distribution. In (a), curves for *Algo A* 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 lower bounds

Figure 10: *Overall mean access time* for different values of access skew coefficient θ and using random length distribution. In (a), curves for *Algo A* and *optimal* overlap with each other. The simulation results are within 0.3% of analytical results.

Figures 7, 8, 9 and 10 plot *overall mean access time* for different values *of access skew coefficient* θ, for the four length distributions presented earlier. In each of these figures, part (a) plots the simulation results, and (b) plots the analytical lower bound on *overall mean access time*. In part (a), the curve labeled "Algo A" corresponds to the access time obtained by simulating algorithm A. The curves labeled "i buckets" in part (a) correspond to the access time obtained by simulating algorithm B with i buckets. In part (a) and (b) both, the curve labeled "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 with i buckets.

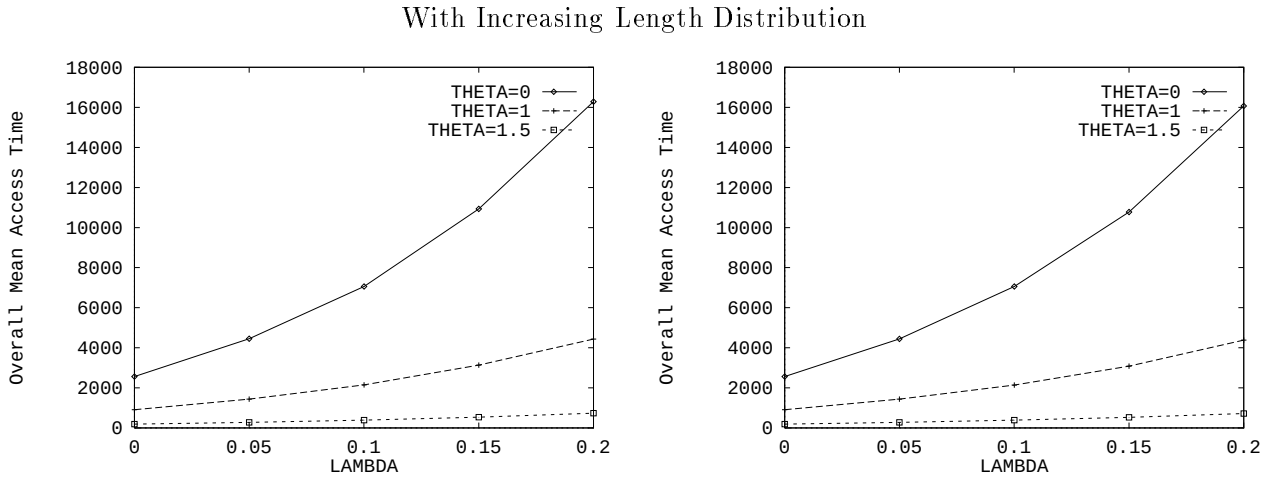
First observation, as noted in the caption of each figure, is that the simulation results are very close to the corresponding lower bounds obtained analytically. Thus, our algorithm performs well (close to optimal). Now note that, when number of buckets is 1, Algorithm B reduces to the so called "flat" 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. 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 a reasonably small 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 lower bounds, analytical bounds 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

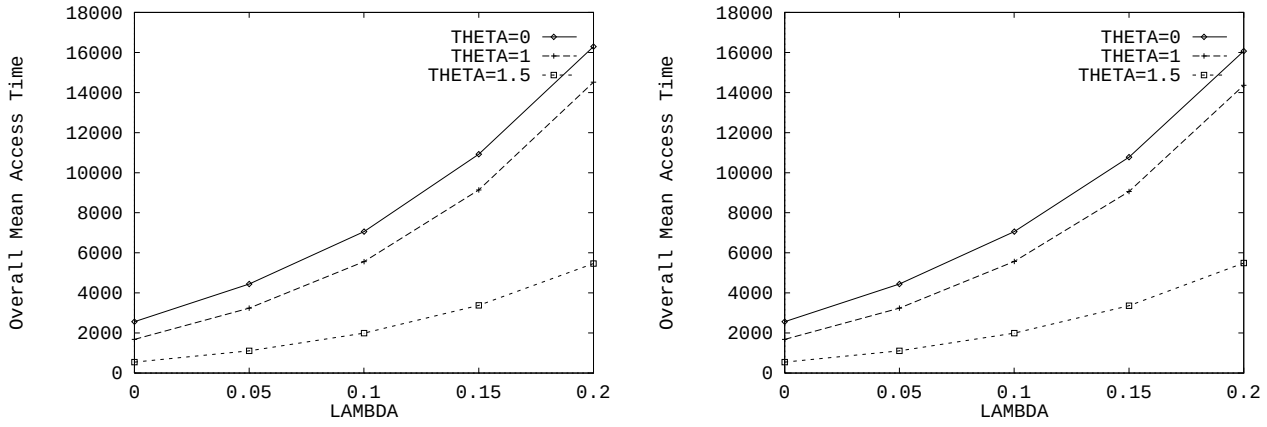
In this section, we evaluate performance of the on-line algorithm in the presence of uncorrectable errors as explained in section 4. For the sake of illustration, we assume that uncorrectable errors occur according to a Poisson process with rate λ. Hence $E(l_i) = 1 - e^{-\lambda l_i}$. Figures 11 and 12 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 lower bounds for $\theta = 0, 1$ and 1.5. The lower bounds 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. 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*. Note that, with uniform length distribution, the term $(1+E(l_i))/(1-E(l_i))$ becomes a constant (independent of i). Therefore, for uniform length distribution, Theorem 2 reduces to Theorem 1.



(a) Simulation results (b) Analytical lower bounds

Figure 11: *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.



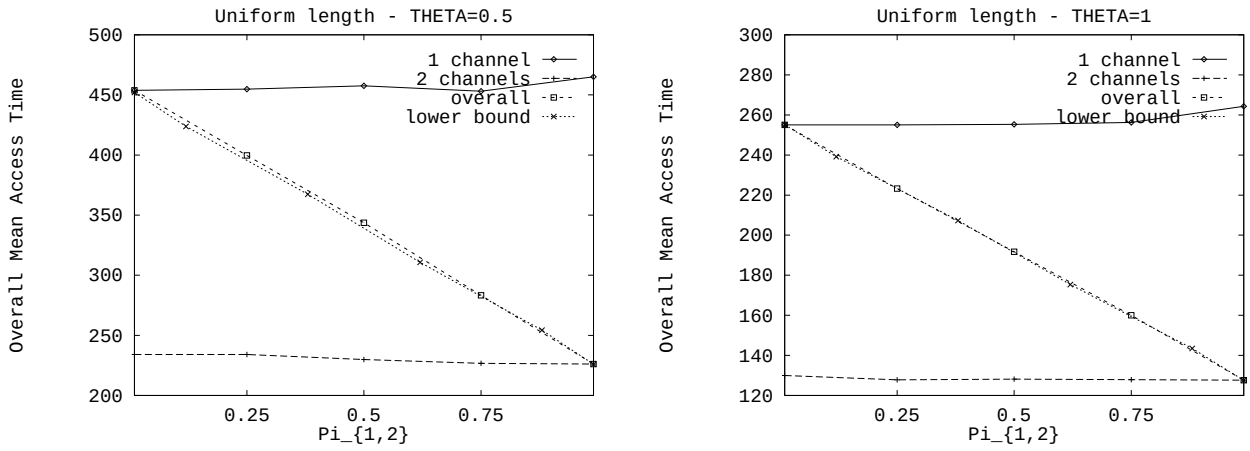
(a) Simulation results (b) Analytical lower bounds

Figure 12: *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.

6.6 Performance with Multiple Broadcast Channels

In this section, we evaluate the performance of the *on-line* algorithm given in Section 5 for multiple channel broadcast with number of channels $c = 2$. We also assume that $\Pi_{\{1\}} = \Pi_{\{2\}} = \frac{1-\Pi_{\{1,2\}}}{2}$.

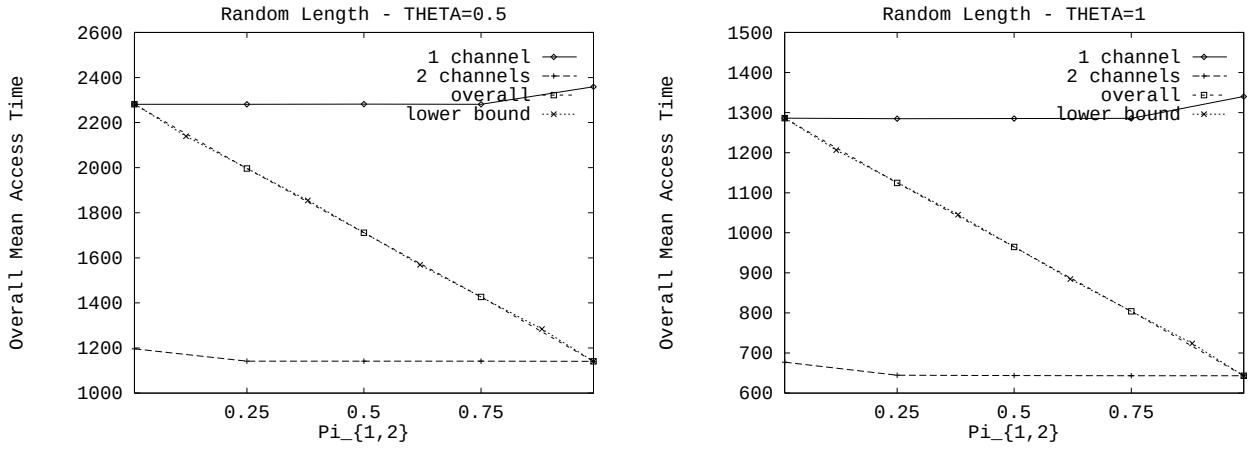
With Uniform Length Distribution



(a) $\theta = 0.5$ (b) $\theta = 1$

Figure 13: Performance Evaluation for Uniform Length Distribution. (a) shows the curves for $\theta = 0.5$ and (b) shows the curves for $\theta = 1$. Note that the overall performance is very close to optimal. The curves labeled "overall" and "lower bound" almost overlap.

With Random Length Distribution



(a) $\theta = 0.5$ (b) $\theta = 1$

Figure 14: Performance Evaluation for Random Length Distribution. (a) shows the curves for $\theta = 0.5$ and (b) shows the curves for $\theta = 1$. Note that the overall performance is very close to optimal. The curves labeled "overall" and "lower bound" almost overlap.

Figures 13 (a) and 13 (b) show the evaluation results for uniform length distribution with

skew coefficient $\theta = 0.5$ and 1 respectively, against different values of $\Pi_{\{1,2\}}$. Figures 14 (a) and 14 (b) show the similar results but for random length distribution with $\theta = 0.5$ and 1 respectively. The curve labeled "1 channel" in each of these figures is obtained for the clients listening to channel 1 (i.e., $t_{\{1\}}$). As $\Pi_{\{1\}} = \Pi_{\{2\}}$, due to symmetry, $t_{\{1\}}$ and $t_{\{2\}}$ should be identical. The curve labeled "2 channels" is obtained for the client listening to both the channels and represents $t_{\{1,2\}}$. The curve labeled "overall" is the *overall mean access time* obtained by substituting the values of $t_{\{1\}}, t_{\{2\}}$ and $t_{\{1,2\}}$ in Equation 9. Whereas the curve labeled "lower bound" is obtained from Equation 10 in Section 5.

For $\Pi_{\{1,2\}} \leq 0.75$, the performance viewed by a client listening to only one channel is very close to optimal obtained by Equation 3. However, as $\Pi_{\{1,2\}}$ approaches to 1 (more and more clients listen to both channels), the *overall mean access time* observed by client listening to only one channel increases. This is because of the fact that for $\Pi_{\{1,2\}}$ approaching to 1, the values of $\Pi_{\{1\}}$ and $\Pi_{\{2\}}$ drop down to such an extent that the terms corresponding to $\Pi_{\{1\}}$ and $\Pi_{\{2\}}$ in the cost function (Equation 11 in section 5) become negligible. Hence the clients listening to only one channel pay the penalty.

The figures also show that for the clients listening to both the channels, the algorithm improves the performance by approximately a factor of 2 (as compared to client listening to one channel). As $\Pi_{\{1,2\}}$ approaches to zero (less number of clients listening to both the channels as compared to total population), the algorithm favors those who are listening to only one channel to improve the *overall mean access time*, penalizing those who are listening to both the channels. The figures also show that the overall performance shown by curve "overall" is quite close to the analytical lower bound shown by curve "lower bound".

For brevity, we do not show the results for increasing and decreasing length distribution.

7 Related Work

The problem of data broadcasting has received much attention lately. The existing schemes can be roughly divided into two categories (some schemes may actually belong to both categories): 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 A 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 broadcast frequencies. 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 report 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).

Battlefield Awareness and Data Dissemination (BADD) Advanced Concept Technology Demonstration (ACTD) is a project in which our research work may be applied [13]. ACTD is managed and funded by DARPA Information System Services. The mission behind BADD project is to develop an operational system that would allow information dissemination in battlefields, maintain access to worldwide data repositories and provide tools to dynamically tailor the information system to changing battlefield situations in order to allow warfighters to view a consistent picture of the battlefield.

8 Summary

This report 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 report are as follows:

- We propose 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 report facilitates a trade-off between time complexity and performance of the on-line algorithm.
- The report considers the impact of errors on optimal broadcast schedules. A near-optimal algorithm for scheduling in presence of errors is presented.
- 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. The clients listening to multiple channels should experience proportionately lower delays. This report presents an algorithm for scheduling broadcasts in such a system.

More work is needed on some problems discussed in this report. 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 at

`http://www.cs.tamu.edu/faculty/vaidya/mobile.html` .

Acknowledgements

We thank referees of the WOSBIS workshop version of this paper [16] for their valuable comments and suggestions.

A Appendix: Proof of Theorem 1

Proof: As instances of item i are spaced equally, the spacing between consecutive instances of item i is N/f_i, where $N = \sum_{j=1}^{M} f_j l_j$ is the length of the broadcast cycle. Also, in this case, the *item mean access time* is $t_i = s_i/2$. Therefore, $t_i = \frac{N}{2f_i}$. Now, *overall mean access time* $t = \sum_{i=1}^{M} p_i t_i$. Therefore, we have,

$$t = \sum_{i=1}^{M} p_i \frac{N}{2f_i} = \frac{1}{2} \sum_{i=1}^{M} p_i \frac{N}{f_i} \tag{12}$$

Define "supply" of item i, $r_i = \frac{f_i l_i}{N}$. Thus, r_i is the fraction of time during which item i is broadcast. Now note that, $\sum_{i=1}^{M} r_i = \sum_{i=1}^{M} \frac{f_i l_i}{N} = \frac{N}{N} = 1$. Now, Equation 12 can be rewritten as,

$$t = \frac{1}{2} \sum_{i=1}^{M} \frac{p_i l_i}{r_i} \tag{13}$$

As $\sum_{i=1}^{M} r_i = 1$, only $M-1$ of the r_i's can be changed independently. Now, for the optimal values of r_i, we must have $\frac{\partial t}{\partial r_i} = 0, \forall i$. We now solve these equations, beginning with $0 = \frac{\partial t}{\partial r_1}$.

$$\begin{aligned}
0 &= \frac{\partial t}{\partial r_1} = \frac{1}{2} \frac{\partial}{\partial r_1} \left(\sum_{i=1}^{M} \frac{p_i l_i}{r_i} \right) \\
&= \frac{1}{2} \frac{\partial}{\partial r_1} \left(\frac{p_1 l_1}{r_1} + \sum_{i=2}^{M-1} \frac{p_i l_i}{r_i} + \frac{p_M l_M}{(1 - \sum_{i=1}^{M-1} r_i)} \right) = \frac{1}{2} \left(-\frac{p_1 l_1}{r_1^2} + \frac{p_M l_M}{(1 - \sum_{i=1}^{M-1} r_i)^2} \right) \\
\Longrightarrow \quad \frac{p_1 l_1}{r_1^2} &= \frac{p_M l_M}{(1 - \sum_{i=1}^{M-1} r_i)^2} \qquad (14) \\
\text{Similarly} \quad \frac{p_2 l_2}{r_2^2} &= \frac{p_M l_M}{(1 - \sum_{i=1}^{M-1} r_i)^2} \qquad (15) \\
& \qquad (16)
\end{aligned}$$

From Equations 15 and 16, we get

$$\begin{aligned}
\frac{p_1 l_1}{r_1^2} = \frac{p_2 l_2}{r_2^2} \Longrightarrow \frac{r_1}{r_2} &= \sqrt{\frac{p_1 l_1}{p_2 l_2}} \\
\text{Similarly it can be shown that} \quad \frac{r_i}{r_j} &= \sqrt{\frac{p_i l_i}{p_j l_j}}, \quad \forall i, j
\end{aligned}$$

This implies that, the optimal r_i must be linearly proportional to $\sqrt{p_i l_i}$. It is easy to see that constant of proportionality $a = \frac{1}{\sum_{j=1}^{M} \sqrt{p_j l_j}}$ exists such that $r_i = a\sqrt{p_i l_i}$ is the *only* possible solution for the equations $\frac{\partial t}{\partial r_i} = 0$. From physical description of the problem, we know that a non-negative minimum of t must exist. Therefore, the above solution is unique and yields the minimum t.

Substituting $r_i = \frac{\sqrt{p_i l_i}}{\sum_{j=1}^{M} \sqrt{p_j l_j}}$ into Equation 13, and simplifying, yields optimal *overall mean access time* as $t = \frac{1}{2}\left(\sum_{i=1}^{M} \sqrt{p_i l_j}\right)^2$. Also, the optimal frequency of item i, f_i may be obtained as $f_i = \frac{r_i N}{l_i} \propto \sqrt{p_i l_i}\frac{N}{l_i} = \sqrt{\frac{p_i}{l_i}}N$. Thus, we have shown that, optimal frequency f_i is directly proportional to $\sqrt{\frac{p_i}{l_i}}$. □

B Appendix: Optimal Overall Mean Access Time with Bucketing

Proof: Bucket B_j $(1 \leq j \leq k)$ contains m_j items, such that $\sum_{j=1}^{k} m_j = M$. Also, $q_j = (\sum_{i \epsilon B_j} p_i)/m_j$ and $d_j = (\sum_{i \epsilon B_j} l_i)/m_j$ be the average access probability and average length of the items in bucket B_j, respectively.

The proof here is similar to the proof in Appendix A. With bucketing, the frequency of all items in the same bucket is identical. We define F_i as the frequency of items in bucket B_i. For optimal solution, the items should be equally spaced. Therefore, spacing between consecutive instances of an item in bucket i is N/F_i. Let S_i denote the spacing N/F_i.

Now, $N = \sum_{j=1}^{k} F_j d_j m_j$. Therefore, $S_i = N/F_i = \sum_{j=1}^{k} F_j d_j m_j / F_i$. Let T_i denote the *item mean access time* of an item in bucket B_i. Then, $T_i = \frac{1}{2}S_i = \frac{1}{2}N/F_i = \frac{1}{2}(\sum_{j=1}^{k} F_j d_j m_j)/F_i$. Note that, with the equal spacing assumption, *item mean access time* is identical for all items in the same bucket.

The *Overall Mean Access Time* is now given by

$$t = \sum_{j=1}^{k} \left(\sum_{i \in B_j} p_i \right) T_j$$

Since $\sum_{i \in B_j} p_i = m_j q_j$, the above equation can be written as $t = \sum_{j=1}^{k} m_j q_j T_j$ or $t = \frac{N}{2} \sum_{j=1}^{k} \frac{q_j m_j}{F_j}$.

We define *supply* of bucket B_j, denoted r_j, as $r_j = F_j d_j m_j / N$. Observe that $\sum_{j=1}^{k} r_j = 1$. The above equation for t can be rewritten as

$$t = \frac{1}{2} \sum_{j=1}^{k} \frac{q_j m_j^2 d_j}{r_j} \tag{17}$$

If we denote $q_j m_j^2 d_j$ as X_j, the above equation becomes

$$t = \frac{1}{2} \sum_{j=1}^{k} \frac{X_j}{r_j}$$

where $\sum_{j=1}^{k} r_j = 1$. This equation has the same form as Equation 13. Therefore, from the proof in Appendix A it follows that, with bucketing, to minimize t the following condition must be true:

$$r_j \propto \sqrt{X_j} \tag{18}$$

As $\sum_{j=1}^{k} r_j = 1$, $r_j = \frac{\sqrt{X_j}}{\sum_{i=1}^{M} \sqrt{X_j}}$. Substituting this into Equation 17, replacing $X_j = q_j m_j^2 d_j$, and simplifying, we get

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

Substituting $X_j = q_j m_j^2 d_j$ in the above proportionality (18), we get

$$r_j \propto \sqrt{q_j m_j^2 d_j} = m_j \sqrt{q_j} \sqrt{d_j}$$

As $r_j = F_j d_j m_j / N$, we now get $\frac{F_j d_j m_j}{N} \propto m_j \sqrt{q_j} \sqrt{d_j}$. On simplifying, this yields, $F_j \propto \sqrt{q_j / d_j}$. As $F_j = N/S_j$, we have, $S_j \propto \sqrt{d_j / q_j}$. Finally, note that, if item i is in bucket B_j, then its frequency $f_i = F_j$ and its spacing $s_i = S_j$.

□

C Appendix: Overall Mean Access Time in Presence of Errors

Here we are not assuming bucketing. The result below can be easily generalized to the case where items are divided into buckets. Consider item i, instances of which are spaced s_i time units apart. The total time required to transmit the cycle is N. Then, $f_i = N/s_i$. Also, as size of item i is l_i, we have $\sum_{i=1}^{M} f_i l_i = N$.

First, let us determine the *item mean access time*, t_i, for item i. Observe that *average* time until the first instance of item i is transmitted, from the time when a client starts waiting for item i, is $s_i/2$ time units. If the first instance of item i transmitted after a client starts waiting is corrupted, then an additional s_i time units of wait is needed until the next instance. Thus, each instance of item i that is received with uncorrectable errors adds s_i to the access time. Given that the probability that an instance of item i of length l_i contains uncorrectable errors is $E(l_i)$, the expected number of consecutive instances with uncorrectable errors is obtained as

$$\frac{E(l_i)}{1 - E(l_i)}$$

Thus, the *item mean access time* is obtained to be

$$t_i = \frac{s_i}{2} + s_i \left(\frac{E(l_i)}{1 - E(l_i)} \right) = s_i \left(\frac{1}{2} + \frac{E(l_i)}{1 - E(l_i)} \right) = \frac{1}{2} s_i \left(\frac{1 + E(l_i)}{1 - E(l_i)} \right)$$

Therefore,

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

Proof of Theorem 2 : As $s_i = N/f_i$, the above expression for t can be rewritten as,

$$t = \frac{1}{2}\sum_{i=1}^{M} p_i \frac{N}{f_i}\left(\frac{1+E(l_i)}{1-E(l_i)}\right) = \frac{1}{2}\sum_{i=1}^{M} \frac{p_i l_i \left(\frac{1+E(l_i)}{1-E(l_i)}\right)}{r_i}$$

where, $r_i = f_i\, l_i/N$. Now, $\sum_{i=1}^{M} r_i = \sum_{i=1}^{M} f_i l_i/N = N/N = 1$. Thus, the above expression for t has a form similar to Equation 13, and can be minimized similarly. The optimization procedure yields the result stated in Theorem 2.

References

[1] S. Acharya, M. Franklin, and S. Zdonik, "Prefetching from a broadcast disk," in *12th International Conference 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," May 1994.

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

[12] R. Jain and J. Werth, "Airdisks and airRAID : Modelling and scheduling periodic wireless data broadcast (extended abstract)," Tech. Rep. DIMACS Tech. Report 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, Computer Science Dept., Texas A&M University, May 1996.

[15] N. H. Vaidya and S. Hameed, "Data Broadcast Scheduling: On-line and Off-line Algorithms," Tech. Report 96-017, Computer Science Dept., Texas A&M University, May 1996.

[16] N. H. Vaidya and S. Hameed, "Data Broadcast in Asymmetric Wireless Environments," in *Workshop on Satellite Based Information Services (WOSBIS)*, Nov. 96.

[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.