Data Broadcast Scheduling*
(Part 1)

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 96-012

May 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 two algorithms for determining broadcast schedules that minimize the wait time. Simulation results are presented to demonstrate that our algorithms perform well.

*Research reported is supported in part by Texas Advanced Technology Program. grant 009741-052-C.

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. For instance, such mechanisms could be used by a mobile support station (or base station) to communicate information of common interest (e.g., stock quotes, sports scores, etc.) to the mobile hosts within its cell. Approaches for determining what to transmit and when, is the subject of this report.

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* [1-3,5-13]. 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 [5, 10, 11, 6, 3, 2, 13]
- Tuning time: This is the amount ot 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. Previous work has tried to minimize the tuning time by providing the clients with an *index* that provides hints to the client of when the required data is scheduled to be broadcast [8, 10, 12]. Without this information, the client must listen continually until the necessary information is received.

This report presents an approach to minimize the *access time*. Our results can be combined with the schemes for reducing *tuning time* to obtain a comprehensive solution. We consider a database that is divided into *pages*. Thus, a broadcast schedule specifies when each page is to be transmitted.

The main contributions of this report are as follows:

- **Square-root Rule:** We show that the access time is minimized when the frequency of a page (in the broadcast schedule) is proportional to the *square-root* of its demand

(characterized as *demand probability*). While this rule is similar to a result by Imielinski and Viswanathan [9] for a *probabilistic* scheduling algorithm, our approach results in half the access time as compared to that in [9]. Our result is derived for a deterministic algorithm, based on a somewhat different model. The similarity between our result and that in [9] suggests that the square-root rule may be inherent to optimal scheduling of broadcast schemes.

- We present two algorithms for scheduling broadcasts, based on the above *square-root rule*. One algorithm generates a cyclic schedule for a *given* cycle size. The second algorithm is not constrained to produce a cyclic schedule with any specified size. Each time a data page is to be transmitted, this algorithm is called, and it returns the identifier of the page that should be transmitted. Thus, the first algorithm can be considered off-line, while the second algorithm is on-line. We discuss relative merits of the two algorithms later. It is worth noting that these algorithms can be combined with the ideas previously proposed in [9, 10, 12] to minimize access time as well as tuning time.

  Simulation results are presented to demonstrate the performance of our algorithms.

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

The rest of the report is organized as follows. Section 2 introduces some terminology. Section 3 describes two types of approaches for scheduling broadcast cycles: static and adaptive. The focus of this report is on static scheduling; we briefly describe how our schemes can be made adaptive. Section 4 derives the *square-root* rule, and presents our static broadcast scheduling algorithms. The impact of errors is analyzed in Section 5. Section 6 evaluates the performance of our schemes. Section 7 briefly discusses how our schemes can be made adaptive. Related work is summarized in Section 8. The report concludes with Section 9.

2 Preliminaries

In this section, we introduce terminology and notation to be used in the rest of the report.

We assume that the database at server is divided into *pages*. Size of a page is measured in terms of the amount of time it takes to broadcast the page. The main results obtained

in this report are applicable when the pages are *not* necessarily identical in size. However, for simplicity, unless otherwise specified all pages are assumed to be identical in size.

In much of our initial discussion, we focus on *cyclic scheduling*, i.e., algorithm to design a *broadcast cycle* schedule with a specified cycle size. We will apply our results to later design an "on-line" algorithm for broadcast scheduling.

The time required to broadcast one page is referred to as one *pagetime*. Let M denote the total number of data pages in the server's database. The broadcast cycle is divided into a fixed number[1] of *slots*; the number of slots being denoted as N. Each broadcast slot contains one page of information. One *pagetime* is equal to the time required to broadcast one slot. An appearance of a page in a broadcast slot is referred to as an *instance* of the page. There may be multiple instances of a page in the cycle. *Schedule* for the broadcast cycle is an assignment of the pages to the slots in the cycle. More formally, a schedule is a function ψ, $\psi : S \rightarrow D$, where $D = \{d | 1 \leq d \leq M\}$ represents the set of M data pages and $S = \{s | 0 \leq s \leq N-1\}$ represents the set of slots in the broadcast cycle.

We define the *frequency* of a page as the number of instances of the page in the broadcast cycle. f_i denotes frequency of page i. The f_i instances of a page are numbered 1 through f_i. The *spacing* between two instances of a page scheduled in slots A and B ($B > A$) is equal to ($B - A$) modulo N. s_{ij} denotes the spacing between j-th instance of page i and the next instance of page i ($1 \leq j \leq f_i$). Note that, after the f_i-th instance of a page in a transmission of the broadcast cycle, the next instance of the same page is the *first* instance in the next transmission of the broadcast cycle.

Page Mean Access Time of a page i, denoted t_i, is defined as the average wait by a client needing page i until page i is received from the server. Provided that the time at which a client needs a page i is uniformly distributed over the length of the cycle, 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 page 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,

$$s_i = \frac{N}{f_i}.$$

In this case, the expression for t_i can be simplified as follows:

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

[1]It is possible to conceive implementations with varying cycle size. Such schemes are a subject of our on-going research.

$$= \frac{1}{2} f_i \left(\frac{s_i^2}{N}\right) = \frac{1}{2} s_i \text{ as } s_i = N/f_i \quad (1)$$

Let p_i denote the probability that a page needed by a client is page i. Thus, p_i is the *demand probability* for page i. *Overall Mean Access Time*, t, is defined as the average wait encountered by a client (average over all pages). Thus,

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

When $s_{ij} = s_i$ $(1 \leq j \leq f_i)$, that is, when all pages are distributed in the cycle with equal spacing, the above equation reduces to

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

3 Static and Adaptive Broadcasts

In the asymmetric communication environment under consideration, the clients may not be capable of sending requests for data to the server (or, it may not be desirable for the clients to send the requests to the server). In this case, the server must be provided with information (such as user profiles) which will allow the server to estimate the demand probability for each page. The broadcast schedule used by the server is determined completely by this estimate of the demand. The broadcast schedule will not change unless the user profiles are updated. In this report, such broadcast schedules are said to be *static*.

An *adaptive* scheme will continually gather statistics on the demand for various pages, and modify the broadcast schedule to best meet the demand. While our focus is on *static* schemes, our schemes can be made *adaptive*, as discussed in Section 7.

4 Proposed Scheduling Schemes

We first consider cyclic broadcast schedules. Fig. 1 depicts our procedure for constructing a broadcast cycle. The first block in Fig. 1 maps the demand probability distribution into page frequencies. Recall that frequency of a page is the number of times the page is to be broadcast in a cycle. Having determined the frequencies, second block in Fig. 1 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.

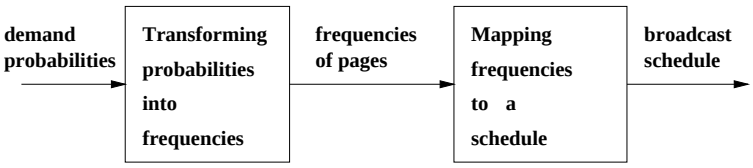

Figure 1: Constructing a Broadcast Cycle

4.1 Mapping Demand to 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 [11], and has been implicitly used by others (e.g., [3]).

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

This result remains valid when all pages in the server database are not identical in size.

Proof of the lemma is omitted here for brevity. Clearly, it is not always possible to space instances of a page equally. For example, if number of slots in the cycle is 100 and frequency for page 1 is 15, the instances of page 1 cannot be spaced evenly, as 100/15 is not an integer. However, the above simple observation provides a heuristic, which is used by our algorithm for scheduling the broadcast cycle. Note that, while Lemma 1 suggests that spacing between consecutive instances of page i should be constant, say s_i, s_i need not be identical to the spacing s_j between instances of another page 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). We assume the ideal situation, as implied by Lemma 1, where instances of all pages can be equally spaced. This assumption, although often difficult to implement, does lead to a useful result. Before presenting the result formally, it is instructive to consider the two examples below.

An intuitive (but not necessarily optimal) way to map demand probabilities into frequencies is to choose the frequencies to be linearly proportional to the probabilities. That is,

$$f_i \quad \propto \quad p_i, \text{ for } 1 \leq i \leq M$$

The example below presents an illustration.

Example 1: Suppose there are 5 pages in the server's database. Their demand probabilities are given below :

page number i	1	2	3	4	5
demand probability p_i	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$

Taking frequencies linearly proportional to demand probabilities, for $N = 8$, page frequencies in the table below are obtained. Note that the frequency of page 1 is four times that of page 2, because $p_1 = 4\,p_2$. The table also shows the spacing between consecutive instances of each page. The resulting schedule for the broadcast cycle is illustrated in Fig. 2.

page number i	1	2	3	4	5
frequency f_i	4	1	1	1	1
spacing $s_i = \frac{N}{f_i} = \frac{8}{f_i}$	2	8	8	8	8

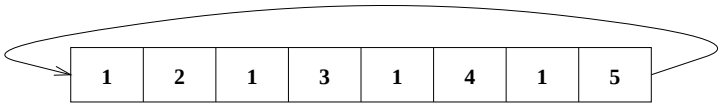

Figure 2: Schedule for Example 1

The *page mean access time* for page i, t_i, can be calculated using Equation 1, as summarized in the table below. The *overall mean access time*, t, calculated using t_i's and Equation 2, is obtained to be equal to $t = 2.5$ *pagetimes*. In general, it can be easily shown that, if all instances of a page are equally spaced, and page frequencies are linearly proportional to demand probabilities, then the *overall mean access time* is $M/2$, where M is the number of pages in the server database. In Example 1, $M = 5$, therefore $t = 5/2 = 2.5$.

page number i	1	2	3	4	5
t_i (unit is *pagetime*)	1	4	4	4	4

Example 2 : For the same demand probabilities as in Example 1, let us now use the frequencies listed in the table below. Note that in this case, $N = 6$. The resulting broadcast schedule is shown in Fig. 3.

page number i	1	2	3	4	5
frequency f_i	2	1	1	1	1
spacing $s_i = \frac{N}{f_i} = \frac{6}{f_i}$	3	6	6	6	6

Using Equations 1 and 2, the *overall mean access time* for the above schedule is obtained as $t = 2.25$ *pagetimes*, which is smaller than $t = 2.5$ obtained for the frequency assignment in Example 1. Note that the frequencies f_i assigned in Example 2 are linearly proportional to $\sqrt{p_i}$. We now show that the minimum possible *overall mean access time* is

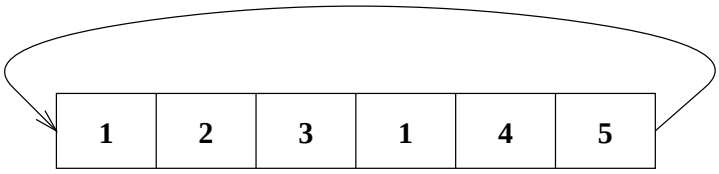

Figure 3: Schedule for Example 2

achieved when

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

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

Proof: Appendix B presents the proof. □

It is worth noting that Theorem 1 is applicable even if the pages are not identical in size. As $\sum_{j=1}^{M} f_j = N$, above theorem implies that, $f_i = N\,(\sqrt{p_i}/\sum_{j=1}^{M} \sqrt{p_j})$. Also, as $s_i = N/f_i$, a consequence of the above result is that, for t to be minimized, we need

$$s_i \propto p_i^{-1/2}$$

As seen later, it is hard to satisfy the above condition for many probability distributions. The above result, however, can be used to design good scheduling algorithms. Under the condition specified in Theorem 1, the optimal *overall mean access time*, named t_{optimal}, (derived in Appendix B) is obtained as

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

Relation to Previous Work

Imielinski and Viswanathan [7] previously proposed an algorithm for *publishing* that uses a *probabilistic* approach for deciding which page to transmit in each slot. In their scheme, page i is transmitted in a given slot with probability

$$\frac{\sqrt{p_i}}{\sum_{j=1}^{M} \sqrt{p_j}}$$

The page to be transmitted is determined using a random number generator with the above probability distribution. This probability function does result in page frequencies f_i that are proportional to $\sqrt{p_i}$, as recommended by our square-root rule. However, the *overall mean access time* obtained using our approach is half that obtained with the probabilistic approach in [7]. Specifically, *overall mean access time* obtained using the probabilistic approach in [7] is

$$t_{\text{prob}} = \left(\sum_{i=1}^{M} \sqrt{p_i} \right)^2$$

The difference in the performance of the two algorithm arises, primarily, from the fact that we attempt to equally space consecutive instances of a page, while [7] uses a probabilistic approach. We choose the deterministic approach as it can provide in a factor of 2 improvement over [7].

Now we present the two proposed algorithms. The first algorithm produces cyclic schedules for a specified N, while the second algorithm is not required to produce a cycle with any specified N.

4.2 Cyclic Scheduling Algorithm

Theorem 1 requires that $f_i \propto \sqrt{p_i}$, which in turn implies that $f_i = N \sqrt{p_i} / \sum_{j=1}^{M} \sqrt{p_j}$. Clearly, f_i obtained by this expression may not be an integer, and therefore these frequencies may not be achievable in practice. The result stated in Theorem 1, however, can be used as a heuristic for determining suitable page frequencies. Our procedure for determining page frequencies approximates the above ideal frequencies.

We assume that $N \geq M$, so that each page can be transmitted at least once each cycle. Let n_i denote the desired frequency for page i, that is,

$$n_i = N \frac{\sqrt{p_i}}{\sum_{j=1}^{M} \sqrt{p_j}}$$

and, let f_i denote the actual frequencies assigned to each page. We would like n_i and f_i to be as close as possible, if not equal. As n_i may be non-integer, it is not always possible to obtain $f_i = n_i$. The simple solution of using $f_i = \text{round}(n_i)$ does not work as, in this case, it may not satisfy the requirement $\sum_{j=1}^{M} f_j = N$. Our procedure for *frequency assignment* assigns values of f_i that are close to n_i, while making sure that $\sum_{j=1}^{M} f_j = N$. This procedure is summarized in Appendix A.

Once the desired page frequencies (f_i) are known, as stated in Lemma 1, optimal access time is obtained by distributing instances of each page uniformly throughout the

cycle. In order to distribute page i with frequency f_i uniformly, the spacing s_i should be $\frac{N}{f_i}$, where N is the cycle size. If this ratio is not an integer, then it is clearly impossible to space the pages evenly. Unfortunately, even if the ratio $\frac{N}{f_i}$ is an integer, it may not be possible to distribute instances of all the pages evenly. The example below illustrates one such case:

Example 3: Assume that number of pages is 3, and number of slots $N = 6$. The page frequencies f_i and ratio N/f_i are given in the table below. Note that $\frac{N}{f_i}$ is an integer for all i.

page number i	1	2	3
frequency f_i	3	2	1
$s_i = \frac{N}{f_i} = \frac{6}{f_i}$	2	3	6

In this case, an attempt to schedule the cycle quickly shows that, it is impossible to schedule instances of page 1 equally spaced at distance 2, and instances of page 2 equally spaced at distance 3. To do so requires that one instance of page 1 and 2 both be scheduled in the *same* slot ! We refer to this as a "collision".

Our *cyclic scheduling* algorithm can be summarized as follows. It takes as input, the frequencies f_i determined by the *frequency assignment* procedure.

1. Sort the pages in decreasing order of their frequencies. Let the sorted list be named H.

2. For page i at the front of the sorted list H, do the following steps:

   - Assign the first instance of page i to a randomly chosen empty slot in the cycle, say slot number X (assume that the slots in the broadcast cycle are numbered 0 through $N-1$).
   - Thereafter, for each j-th instance of page i ($2 \leq j \leq f_i$), an attempt is made to schedule the j-th instance in slot number $Y = \text{round}(X + (j-1)\,s_i)$ modulo N, where $s_i = N/f_i$. As $s_i = N/f_i$ may not be an integer, *rounding* operation is performed to determine Y. If slot Y is already filled by some page, then the j-th instance of page i is scheduled in an empty slot that is closest to slot Y (modulo N).

3. Remove page i from list H. If list H is not empty, go to step 2.

Section 6 presents numerical results to illustrate the performance of the cyclic scheduling algorithm. Some simple variations on the above steps are possible; discussion of these variants is omitted for brevity.

4.3 On-line Scheduling Algorithm

The first scheduling algorithm described above produces the entire cyclic schedule, before the data is transmitted. On the other hand, the second algorithm can be used on-line to determine the page to be transmitted in each slot, similar to a *probabilistic* algorithm presented in [12]. (Our algorithm is *not* probabilistic, and yields better performance than [12].)

As noted previously, under optimal conditions, the instances of a page are equally spaced with spacing s_i, where

$$s_i \propto p_i^{-1/2}$$

This can be rewritten as

$$s_i^2 \, p_i = \text{constant}$$

The above observation is used in our second algorithm, as presented below. In this case, slots in the *broadcast* are numbered sequentially starting from 1. The slot number is incremented by 1 each time a page is transmitted. The on-line scheduling algorithm is called to determine which page should be transmitted in the next slot, say slot Q. In the following, let $R(j)$ denote the number of the slot in which an instance of page j was most recently transmitted. Initially, $R(j)$ is initialized to 0, for all j.[2] Note that, $R(j)$ is updated whenever page j is transmitted.

ON-LINE algorithm:

broadcast page i in slot Q
  `if`
    $(Q - R(i))^2 \, p_i \;\geq\; (Q - R(j))^2 \, p_j, \quad 1 \leq i, j \leq M$

      (If above condition is true for more
      than one page, any one of those
      pages may be chosen arbitrarily.)

$Q - R(i)$ is the spacing between the current slot, and the slot in which page i was previously transmitted. Note that, the term

$$(Q - R(i))^2 \, p_i$$

is similar to the term $s_i^2 \, p_i$ in the equality

$$s_i^2 \, p_i = \text{constant}$$

[2] The choice of initial value will not affect the mean access time much, unless the broadcast is for a very short time.

presented above. The motivation behind our algorithm is to attempt to achieve this equality. (It should be noted that this equality is not feasible for all demand probability distributions.)

Example 4: Consider a database containing 3 pages such that $p_1 = 1/2$, $p_2 = 3/8$, and $p_3 = 1/8$. Figure 4 shows the pages transmitted by the server recently in slots 94 through 99 of the broadcast (recall that for on-line algorithm, slots in the broadcast are numbered sequentially starting from 1). The above *on-line* algorithm is called to determine the page to be transmitted in slot 100. Thus, $Q = 100$. Also, from Figure 4, observe that $R(1) = 98$, $R(2) = 97$, and $R(3) = 99$. The *on-line* algorithm evaluates the term $(Q - R(j))^2 p_j$ for $j = 1, 2, 3$ as 2.0, 27/8 and 1/8, respectively. As this term is the largest for page 2, page 2 is transmitted in slot 100.

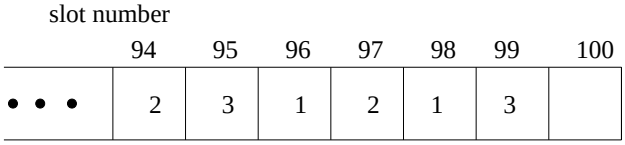

Figure 4: Illustration of the on-line algorithm

It turns out that the above on-line scheduling algorithm also sometime generates cyclic schedules with reasonable size. However, this is not true for all probability distributions.

Performance measurements for the above algorithm are presented in Section 6. It is worth noting that, a similar approach as above can also be used to design cycles of a *given* size N. Evaluation of the above technique when used to design cyclic schedules of a *given* size is a subject of on-going work.

5 Effect of Transmission Errors on Scheduling Strategy

In the discussion above, we assumed that each page 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.

Since in the environment under consideration here we assume that the client does not communicate with the server, it is not possible to ask the server to retransmit the data.[3] In this section, we evaluate the impact of *uncorrectable* errors on the scheduling strategy for broadcasts.

For the purpose of demonstration, we now consider a simple error model. (Similar results can be obtained for other models as well.) Let *uncorrectable* errors occur according to a Poisson process with rate λ per second. Let l_i denote the size of the i-th page, that is, let the time required to broadcast page i be l_i seconds. If all pages have the same size, then $l_i = L$ for all i (L being a constant). Thus, the probability that page i transmitted by the server will be received by a client without uncorrectable errors is $e^{-\lambda l_i}$. If the page contains uncorrectable errors (with probability $1 - e^{-\lambda l_i}$), the page is discarded by the client, and the client cannot use that instance of the page. The client must wait for the next instance of the page. Now let all instances of a page i ($1 \leq i \leq M$) be equally spaced with the spacing of S_i seconds (note: S_i is measured in seconds. s_i defined previously was measured in *pagetime*.)

As shown in Appendix C, under the above assumptions, the following expression for *overall mean access time* (in seconds) is obtained.

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

Using this expression, the result below can be obtained. Appendix C sketches the proof.

Theorem 2 *Given that uncorrectable errors occur according to a Poisson distribution with rate λ, and that all instances of a page are equally spaced, the* overall mean access time *is minimized when*

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

and

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

The above result implies that the frequency of a page should be larger if its size is larger. This is intuitive, because with a larger size, more instances of the page are likely to be corrupted. It is interesting to note that, when all pages are identical in size, the above proportionality reduces to $f_i \propto \sqrt{p_i}$. Thus, for identical sized pages, uncorrectable errors do not have any impact on the optimal scheduling strategy (although the errors do affect the

[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 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.

actual access time). The two algorithms presented in Section 4 can be easily modified to use the result in Theorem 2.

If λ is relatively large, many pages may be corrupted by uncorrectable errors, causing a significant increase in the mean access time. A solution to this problem is to fragment each page into smaller fragments, such that each fragment is encoded separately. Under this situation, a client needs to receive one copy of each fragment so as to be able to receive the page. *However*, this approach requires that the client should be able to receive the fragments out-of-order and reorder them on receipt (the fragments will effectively be received out-of-order if some of the fragments contain uncorrectable errors). Procedure for determining the theoretical optimal frequencies when pages are fragmented to cope with errors is similar to that in the proofs of Theorems 1 and 2.

6 Performance Evaluation

We consider two classes of demand probability distributions:

- Skewed distribution: In this case, half the data pages are requested more frequently than the other half. The distribution is characterized by a parameter B, where $0.5 \leq B \leq 1$. The distribution can be formally defined as below:

$$p_i = \begin{cases} \frac{B}{\lfloor M/2 \rfloor}, & 1 \leq i \leq \lfloor M/2 \rfloor \\ \\ \frac{1-B}{\lceil M/2 \rceil}, & \lceil M/2 \rceil < i \leq M \end{cases}$$

  Varying the value of B yields distributions with different amount of demand "skew".

- Exponential distribution: In this case, page 1 is requested most frequently, and page M is requested least frequently, the probabilities (p_i's) being determined by an *exponential* function of i. This distribution is characterized by a parameter γ. The distribution can be formally defined as below:

$$p_i = p_1 \ e^{-\gamma (i-1)}$$

  where p_1 is evaluated by solving the equality $\sum_{i=1}^{M} p_i = 1$. Thus, $p_1 = (1 - e^{-\gamma})/(1 - e^{-M\gamma})$.

  Larger values of γ cause p_i to decrease more rapidly with increasing i.

We compare the *overall mean access time* t achieved by our two algorithms, with the "optimal" t given by Equation 3

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

and with the overall mean access time achieved using the probabilistic approach by Imielinski and Viswanathan [7], as

$$t_{\text{prob}} = \left(\sum_{i=1}^{M} \sqrt{p_i} \right)^2$$

In the figures to be presented below, the curves corresponding to our cyclic scheduling scheme, and on-line scheduling scheme, are labeled as *cyclic* and *on-line*, respectively. The curves corresponding to t_{optimal} and t_{prob} are labeled *optimal* and *probabilistic*, respectively. The curve labeled $M/2$ plots the value $M/2$. If page frequencies f_i are chosen linearly proportional to p_i, then the minimum possible *overall mean access time* is $M/2$.

It should be noted that the theoretical optimum values of t may not be achievable for all probability distributions. However, the theoretical values do provide a measure to determine how close to optimal our algorithms are.

For the simulation results presented here, number of pages M is 50. Similar results are obtained for larger values as well. The reason for choosing a small number was to enable us to run experiments long enough to obtain simulation results with high confidence. Cycle size N for the cyclic scheduling algorithm was chosen to be 100. Later, we will consider the impact of cycle size on the performance of cyclic scheduling algorithm. Each schedule was simulated long enough to simulate 1 Million requests for each schedule. Uncorrectable transmission errors (as in Section 5) are not considered in our simulation.

For the skewed and exponential probability distributions, Figures 5 and 6 plot the *overall mean access time* as a function of parameters B and γ, that characterize the skewed and exponential distributions, respectively.

When B is close to 0.5, the skewed distribution is not very skewed (i.e., all pages are requested with almost uniform probability). As B approaches 1, requests for half of the pages become more frequent than the other half. When all pages are requested with equal probability ($B = 0.5$), choosing f_i to be proportional to p_i or $\sqrt{p_i}$ both result in the same frequencies. Therefore, in Figure 5, the curves for *optimal* and *M/2* meet at $B = 0.5$. When B is larger than 0.5, the minimum access time achievable by choosing $f_i \propto p_i$ is worse than $f_i \propto \sqrt{p_i}$. In Figure 6, note that our algorithms achieve performance better than $M/2$, and quite close to the theoretical optimal t_{optimal}. In particular, the on-line scheduling algorithm achieves performance almost identical to the optimal.

Similar results are obtained in Figure 6 for the exponential distribution. When $\gamma = 0$, the probability distribution is uniform, and all curves, except *probabilistic*, converge (similar to $B = 0.5$ above). As γ is increased, the performance of on-line scheduling algorithm remains close to optimal, however, that of cyclic scheduling starts degrading.

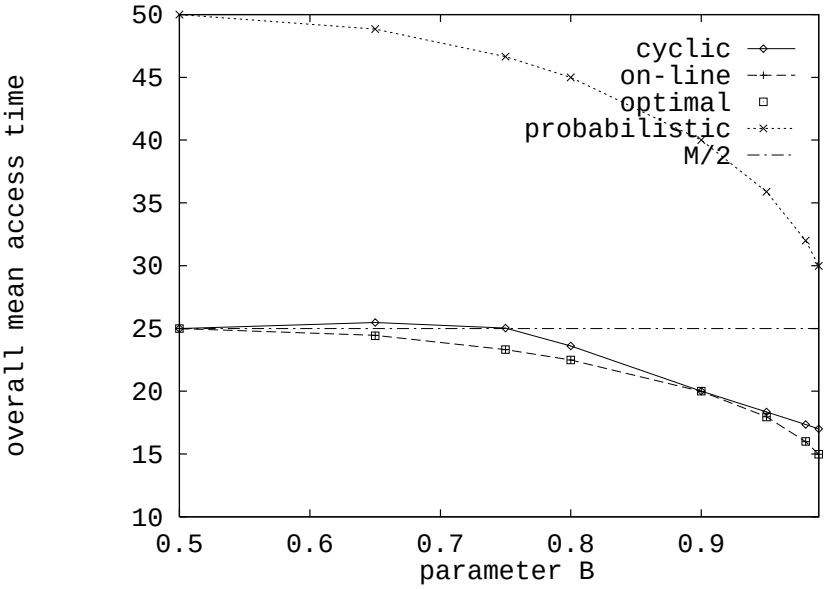

Figure 5: *Overall mean access time* with skewed probability distribution (for different values of B)

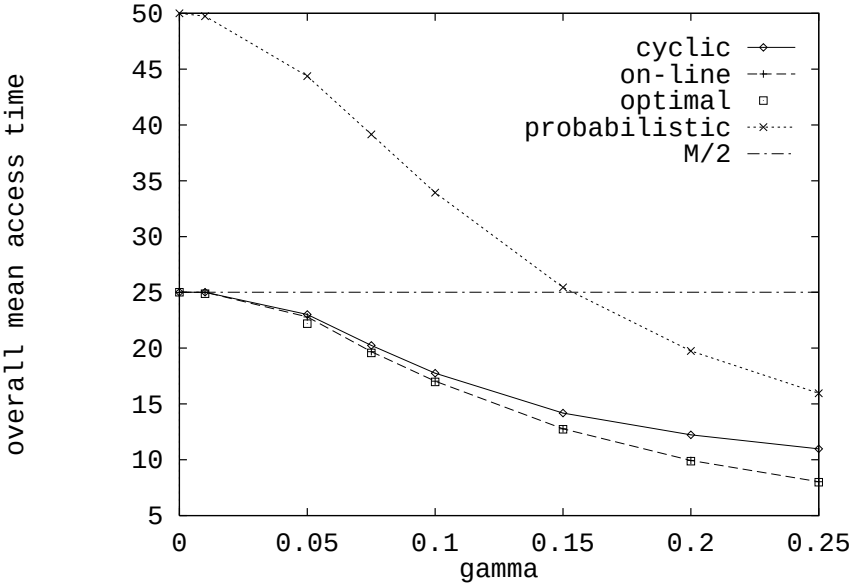

Figure 6: *Overall mean access time* with exponential probability distribution (for different values of γ (gamma))

In general, the *on-line* algorithm performs better than the cyclic scheduling algorithm. The reason for this is that the cyclic scheduling algorithm is constrained to produce a schedule with a specified size ($N = 100$ in our simulations). For many probability distributions, it is not possible to assign frequencies f_i that are (approximately) proportional to $\sqrt{p_i}$, when N is relatively small. For such probability distributions, a cyclic schedule can approach the optimal only when N is large.

To evaluate the impact of the choice of N, for the *cyclic* scheduling algorithm, Figures 7 and 8 plot the overall mean access time, as a function of cycle size N, for selected values of parameters B and γ (γ is denoted as `gamma` in the figures). Captions for the figures also list the theoretical optimum for the overall mean access time. Observe that, for small N (i.e., close to $M = 50$) our cyclic scheduling algorithm is unable to achieve performance close to theoretical optimum. Larger values of N result in performance close to optimal.

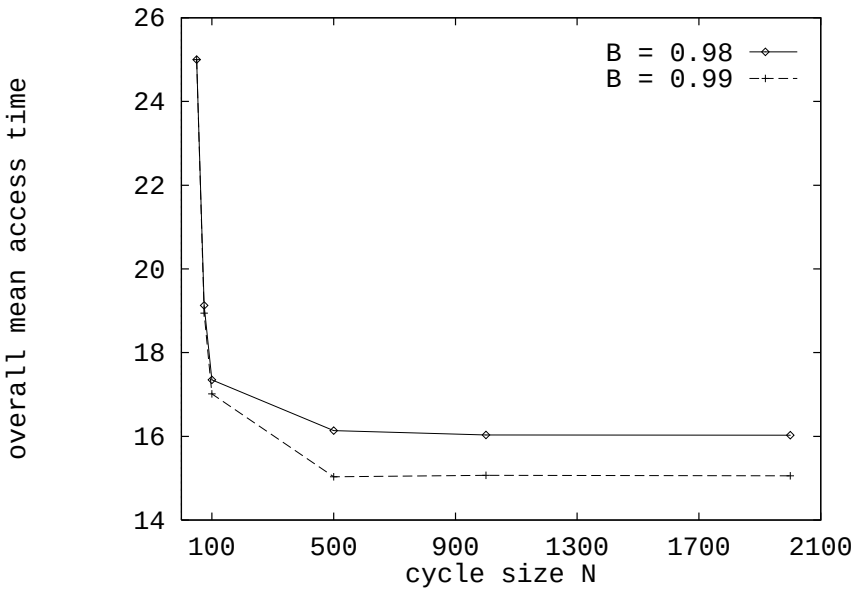

Figure 7: Cyclic schedule with skewed distribution: The theoretical optimum *overall mean access time* for $B = 0.98$ and 0.99 is 16.0 and 14.9875, respectively.

6.1 Merits of the Two Algorithms

The *cyclic scheduling* algorithm produces cyclic schedules of a specified size N. The cyclic schedule can be determined *a priori*, and stored in a look-up table. Thus, cyclic scheduling incurs very little overhead, once the schedule has been determined. In cyclic scheduling, as entire cycle is determined based on the current probability estimates, a schedule change

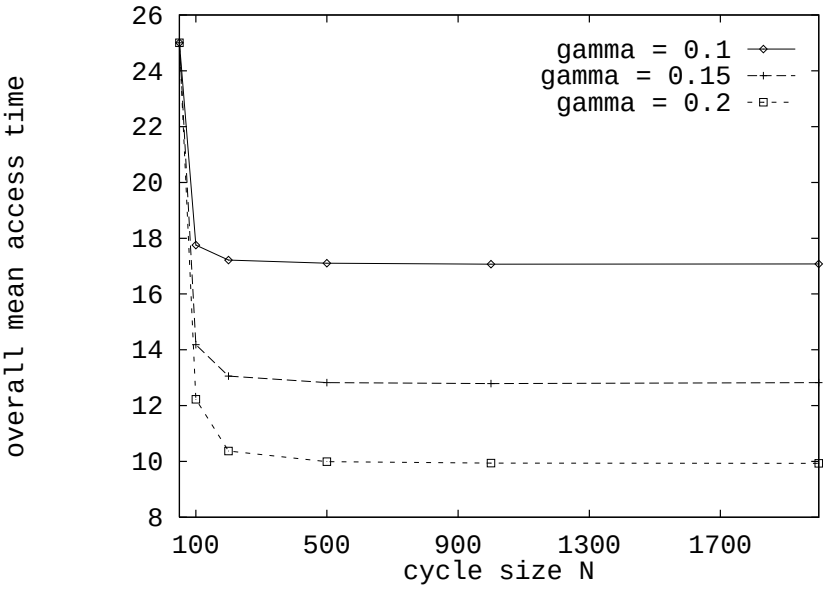

Figure 8: Cyclic schedule with exponential distribution: The theoretical optimum *overall mean access time* for $\gamma = 0.1,\ 0.15$ and 0.2 is 16.9692, 12.7266 and 9.8744, respectively.

may be made only when the next cycle is to be transmitted. Thus, longer broadcast cycles may not be suitable for *adaptive* implementations.

While the on-line algorithm tends to achieve lower access times, it incurs scheduling overhead for each page that is broadcasted. The advantages of the on-line algorithm are (i) lower mean access time, and (ii) ability to quickly adapt to a change in demand probabilities. This is possible, as the page transmitted in each slot can be determined based on current estimate of the demand probabilities.

It may be noted that both our algorithms can be used in conjunction with other techniques for adaptive broadcasts (e.g., [12]).

7 Adaptive Broadcast

To achieve adaptive behavior, the server must have some way to update its estimates of *demand probability* (p_i). Viswanathan [12] describes several ways in which this information can be conveyed to the server, including (i) when a mobile client moves from one cell to another, and (ii) when the client sends some other message, the access frequency information is piggybacked on these messages. The idea is to minimize the need to send messages just to update the user access profiles at the server. Viswanathan [12] divides pages into two classes: (i) pages that are broadcast by the server, and (ii) pages that are sent to clients

only on-demand (i.e., when a request is received from the client for the page). Thus, clients send requests for pages from class (ii), while simply waiting for the server to transmit a page from class (i).

We suggest another approach that does not divide pages into two classes as above. Instead, when a client needs some page, say i, it waits for some time bounded by a *time-out* τ. If an interval of τ is exceeded, then the client sends a request to the server. Thus, if a page is broadcast by the server within τ, no explicit request is sent. Parameter τ affects the mean access time, as well as the number of requests that are sent by the clients. The server, based on the number of requests that are received, can update its estimate of demand probabilities (p_i). These new estimates can be used to determine the appropriate broadcast schedule.[4] Performance evaluation of the above adaptive approach is a subject of on-going research.

8 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* [3, 2, 1, 7, 11, 6, 5, 13] and schemes attempting to reduce the *tuning time* [9, 8, 10]

Chiueh [5] and Acharya et al. [3, 2, 1] present schemes that transmit the more frequently used pages more often. However, they do not use optimal degree of replication, unlike our scheme.

As discussed in Section 4, Imielinski et al. [7] present an adaptive scheme for *publishing* that uses a *probabilistic* approach for deciding which page to transmit in each slot. Their scheme also uses page frequencies proportional to square-root of demand probabilities, however, our approach results in significantly better performance.

Jain and Werth [11] note that reducing the variance of spacing between consecutive instances of a page 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 5, Jain and Werth [11] 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,

[4] In the cyclic scheduling algorithm presented earlier, we ensured that $f_i \geq 1$ for all i. For *adaptive* scheduling, this is not necessary, as a client will eventually time-out for a page whose frequency is 0, and send a request to the server.

much like the RAID approach spreads stripes of data on different disks [4]. This requires the clients to receive broadcasts on multiple frequencies. 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).

9 Conclusions

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. We propose two 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 optimum). The proposed algorithms can be combined with other techniques for minimizing *tuning time* as well as for designing *adaptive* broadcast schedules.

A Appendix: Frequency Assignment Procedure for Cyclic Scheduling

The following steps are performed. Note that $N \geq M$, and we would like to have $f_i \geq 1$, for all i.

1. For $1 \leq i \leq M$, $n_i = N\sqrt{p_i}/\sum_{j=1}^{M}\sqrt{p_j}$. (Determine ideal frequencies.)
2. If $n_i \geq 1$, then $n_i = n_i - 1$, else $n_i = 0$ $(1 \leq i \leq M)$.
3. number of empty slots $= N - M$ (M slots already assigned, one per page.)
4. `balance` $= N - M - \sum_{j=1}^{M} n_i$.
   The total number of slots available is N, of which M slots have already been allocated (one per page). The remaining "demand" for slots is equal to $\sum_{j=1}^{M} n_i$. `balance` denotes the difference between the total number of slots N, and the number of slots needed (which is equal to the number of allocated slots + total unsatisfied "demand"). Therefore, at this point, `balance` $= N - M - \sum_{j=1}^{M} n_i$. Note that, in general, the `balance` may be negative or positive. At the end of this procedure, the `balance` will be 0.

5. The rest of the algorithm only considers those pages for which current value of n_i is greater than 0. These pages (with positive n_i) are ordered according to increasing page numbers (other orderings may also be used).
   Let this ordered list be named G.

6. For the page i at the front of list G, do the following steps.

   - $n_i = n_i +$ `balance`$/|G|$. That is, new n_i is obtained by adding to n_i, page i's "share" of the `balance`.
   - $n_i = \text{round}(n_i)$.
   - If $n_i >$number of empty slots, then $n_i =$ number of empty slots.[5]
   - $f_i = 1 + n_i$. (One slot was reserved for page i in the first two steps of the assignment procedure.)
   - number of empty slots = number of empty slots $- n_i$.
   - remove page i from the front of list G
   - `balance` = number of empty slots $- \sum_{j \in G} n_j$

7. If list G is not empty, go to step 6.

The initial two steps of the above procedure ensure that $f_i \geq 1$, for all i. The remaining steps allocate the remaining slots to different pages, such that the number of allocated slots is close to n_i, while making sure that the sum of assigned frequencies is N. `balance` is used to ensure that frequencies sum to N.

B Appendix: Proof of Theorem 1

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

Proof: As instances of page i are spaced equally, the spacing between consecutive instances of page i is N/f_i, where $N = \sum_{j=1}^{M} f_j$ is the number of slots in the broadcast cycle. Also, in this case, the *page 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,

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

[5] This step can sometime be eliminated without affecting correctness of the algorithm.

Define "supply" of page i, $q_i = \frac{f_i}{N}$. Thus, q_i is the fraction of slots in which page i is broadcast. Now note that, $\sum_{i=1}^{M} q_i = \sum_{i=1}^{M} \frac{f_i}{N} = \frac{N}{N} = 1$. Now,

$$t = \frac{1}{2} \sum_{i=1}^{M} \frac{p_i}{q_i} \tag{4}$$

As $\sum_{i=1}^{M} q_i = 1$, only $M-1$ of the q_i's can be changed independently. Now, for the optimal values of q_i, we must have $\frac{\partial t}{\partial q_i} = 0, \forall i$. We now solve these equations, beginning with $0 = \frac{\partial t}{\partial q_1}$.

$$\begin{aligned}
0 &= \frac{\partial t}{\partial q_1} \\
&= \frac{1}{2} \frac{\partial}{\partial q_1} \left(\frac{p_1}{q_1} + \sum_{i=2}^{M} \frac{p_i}{q_i} \right) \\
&= \frac{1}{2} \frac{\partial}{\partial q_1} \left(\frac{p_1}{q_1} + \sum_{i=2}^{M-1} \frac{p_i}{q_i} + \frac{p_M}{(1 - \sum_{i=1}^{M-1} q_i)} \right) \\
&= \frac{1}{2} \left(-\frac{p_1}{q_1^2} + \frac{p_M}{(1 - \sum_{i=1}^{M-1} q_i)^2} \right) \\
\Longrightarrow \frac{p_1}{q_1^2} &= \frac{p_M}{(1 - \sum_{i=1}^{M-1} q_i)^2} \qquad (5)
\end{aligned}$$

$$\text{Similarly} \quad \frac{p_2}{q_2^2} = \frac{p_M}{(1 - \sum_{i=1}^{M-1} q_i)^2} \tag{6}$$

From Equations 5 and 6, we get

$$\frac{p_1}{q_1^2} = \frac{p_2}{q_2^2} \Longrightarrow \frac{q_1}{q_2} = \sqrt{\frac{p_1}{p_2}}$$

$$\text{Similarly it can be shown that} \quad \frac{q_i}{q_j} = \sqrt{\frac{p_i}{p_j}}, \quad \forall i, j$$

This implies that, the optimal values of q_i's must be linearly proportional to $\sqrt{p_i}$'s. It is easy to see that constant of proportionality $a = \frac{1}{\sum_{j=1}^{M} \sqrt{p_j}}$ exists such that $q_i = a\sqrt{p_i}$ is the *only* possible solution for the equations $\frac{\partial t}{\partial q_i} = 0$. 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 $q_i = \frac{\sqrt{p_i}}{\sum_{j=1}^{M} \sqrt{p_j}}$ into Equation 4, and simplifying, yields optimal *overall mean access time* as

$$t = \frac{1}{2} \left(\sum_{i=1}^{M} \sqrt{p_i} \right)^2$$

□

C Appendix: Overall Mean Access Time in Presence of Errors

Consider page i that is spaced S_i seconds apart. Let the total time required to transmit the cycle be denoted as T. Then, $f_i = T/S_i$. (This is analogous to $f_i = N/s_i$ in case of fixed size pages.) Also, as size of page i is l_i, we have $\sum_{i=1}^{M} f_i l_i = T$. (This is analogous to $\sum_{i=1}^{M} f_i = N$, in case of fixed size pages.)

First, let us determine the *page mean access time* t_i, for page i. While a more formal analysis can be carried out using a Markov chain, here we choose to present an intuitive description of the analysis. Observe that *average* time until the first instance of page i is transmitted, from the time when a client starts waiting for page i, is $S_i/2$ second. If the first instance of page i transmitted after a client starts waiting is corrupted, then an additional S_i seconds of wait is added until the next instance. Thus, each instance of page i that is received with uncorrectable errors adds S_i to the waiting time. Given that the probability that an instance of page i contains uncorrectable errors is $(1 - e^{-\lambda l_i})$, the expected number of consecutive instances with uncorrectable errors is obtained as

$$\frac{(1 - e^{-\lambda l_i})}{1 - (1 - e^{-\lambda l_i})} = \frac{(1 - e^{-\lambda l_i})}{e^{-\lambda l_i}} = e^{\lambda l_i} - 1$$

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

$$t_i \;=\; \frac{S_i}{2} + S_i\,(e^{\lambda l_i} - 1) \;=\; S_i\,\left(e^{\lambda l_i} - \frac{1}{2}\right)$$

Thus,

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

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

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

where, $r_i = f_i\, l_i/T$. Now, $\sum_{i=1}^{M} r_i = \sum_{i=1}^{M} f_i l_i/T = T/T = 1$. Thus, the above expression for t has a form similar to Equation 4, 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] 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.

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

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

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

[8] 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.

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

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

[11] 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.

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

[13] 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.