Scheduling data broadcast in asymmetric communication environments

Nitin H. Vaidya and Sohail Hameed

Department of Computer Science, Texas A&M University, College Station, TX 77843-3112, USA

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

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 wireless clients are of significant interest. 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 [1]. In an asymmetric environment, *broadcasting* the information is an effective way of making the information available simultaneously to a large number of users. When some information is *broadcast*, all pending requests for that information are served simultaneously. For asymmetric environment, researchers have previously proposed algorithms for designing *broadcast schedules* [1,3,4,7–10,12–15,20,21].

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. We present an approach to design broadcast schedules that attempt to minimize the average "access time". *Access time* is the amount of time a client has to wait for an information item 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 [1,4,7,9,14,15,20,21]. Note that, in general, a client may request multiple items simultaneously [5,6,8,12]. In this case, the access time may depend on the number of items requested. Also, the client would expect to receive mutually consistent versions of the requested items. In this paper, similar to some of the past work (e.g., [1,7,9,20]), we consider the simplest case where a client only requests one item per request, and present algorithms to minimize the mean access time. The issue of consistency of items requested in different requests is not considered here.

While *mean access time* is the performance metric considered in this paper, note that other metrics are also relevant in the context of broadcast scheduling. For instance, the user may want to reduce the "tuning" time, i.e., the time for which the user must actively *listen* to the broadcast [13,20]. In other cases, the user may be interested not only in reducing the mean but also the variance of access time. Alternatively, the user may want to minimize the mean access time under the constraint that the worst case access time be limited by a specified upper bound. We consider strategies for reducing the variance of access time elsewhere [16].

In this paper, we also analyze the impact of transmission *errors* on the scheduling policy. 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.

In environments where different clients may listen to different number of broadcast channels (depending on how many they can afford), the schedules on different broadcast channels should be coordinated so as to minimize the access time for most clients. This paper presents an algorithm to minimize access time for clients listening to varying number of channels.

The rest of the paper is organized as follows. Section 2 introduces some terminology. Section 3 presents two broadcast scheduling 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. 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 assumptions to be used in the rest of the paper. Database at the server is assumed to be divided into many *information items*. The items are **not** necessarily of the same size. 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. M denotes the total number of information items in the server's database. The items are numbered 1 through M. l_i represents the length of item i. Arrival of client requests is assumed to be governed by a Poisson process. An appearance of an item in the broadcast is referred to as an *instance* of the item. The *spacing* between two consecutive 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.

If all consecutive instances of an item i are equally spaced, then s_i denotes the spacing for item i.

Item Mean Access Time of item i, denoted as t_i, is defined as the average wait by a client needing item i until it starts receiving item i from the server. If all consecutive instances of item i are equally spaced with spacing s_i, then, from the Poisson process assumption for request arrivals, it follows that [3] $t_i = s_i/2$.

Demand probability p_i denotes the probability that item i is requested in any request. The demand probability is obtained as an average over all clients served by the server. In our discussion, we assume that all items have the same *priority*. When different items have different priorities, it may be more important to keep access times smaller for certain items than others (independent of their demand probability). One potential approach to take priorities into account is to attach *weight* w_i to the access times for item i, according to its priority. When such weights are attached, the analysis below needs to be modified to replace each occurrence of p_i with $(p_i w_i)$. For simplicity, we will assume that all items have the same priority – effectively, w_i is assumed to be 1 for all i.

Overall Mean Access Time, denoted as t, is defined as the average wait encountered by a request. Therefore [3],

$$t = \sum_{i=1}^{M} p_i t_i. \tag{1}$$

When the consecutive instances of item i are spaced s_i apart, $t_i = s_i/2$, therefore, the overall mean access time is given by

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

3. Proposed scheduling schemes

In sections 3 and 4, we consider the case when the information items are broadcast on a single channel. Section 5 considers multiple channel broadcasts. Lemma 1 below states an intuitive observation that follows from a result presented in [15]. This observation has also been implicitly used by others (e.g., [1,4,21]).

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 (denoted as s_i), it need not be identical to the spacing s_j between instances of another item j. Assuming equal spacing s_i for instances of each item i, theorem 1 below states a result obtained by generalizing a result derived in [4,21]. While the result in [4,21] is applicable only to items of identical size, theorem 1 applies to items of differing sizes as well. We use this result to design broadcast scheduling algorithms.

Theorem 1 (Square-root rule). Assuming that instances of each item are equally spaced, minimum *overall mean access time* is achieved when spacing s_i of each item i is proportional to $\sqrt{l_i}$ and inversely proportional to $\sqrt{p_i}$. That is, $s_i \propto \sqrt{l_i/p_i}$.

Appendix A presents proof of the above theorem. As shown in the appendix, when the condition in theorem 1 is satisfied, optimal *overall mean access time*, named t_{optimal}, is obtained as

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

t_{optimal} is derived assuming that instances of each item are equally spaced. As this assumption cannot always be realized, t_{optimal} represents a *lower bound* on achievable overall mean access time. The lower bound, in general, is not achievable. However, as shown later, it is possible to achieve *overall mean access time* almost identical to the above lower bound. Now we present two scheduling algorithms.

3.1. Broadcast scheduling algorithm

Whenever the server is ready to transmit a new item, it calls the algorithm presented here. The algorithm determines the item to be transmitted next using a *decision rule* – this decision rule is motivated by theorem 1. Theorem 1 implies that, for optimal performance, instances of an item i should be equally spaced with spacing s_i, such that

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

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. Note that $R(j)$ is updated whenever item j is transmitted. Let function $G(j)$ be defined as $G(j) = (Q - R(j))^2 p_j / l_j$, $1 \leqslant j \leqslant M$. Our first broadcast scheduling algorithm is named algorithm A. (Note that our algorithms for asymmetric environment can be applied to a *pull*-based broadcast environment, by replacing p_i by the number of pending requests for item i.)

Broadcast scheduling algorithm A

Step 1. Determine maximum value of $G(j)$ over all items j, $1 \leqslant j \leqslant M$. 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 1. 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 1 shows the items recently broadcast by the server (up to time $<$ 100). The above algorithm is called to determine the item to be transmitted at time 100. Thus, $Q = 100$. Also, from figure 1, observe that $R(1) = 95$, $R(2) = 93$, and $R(3) = 96$. The 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.

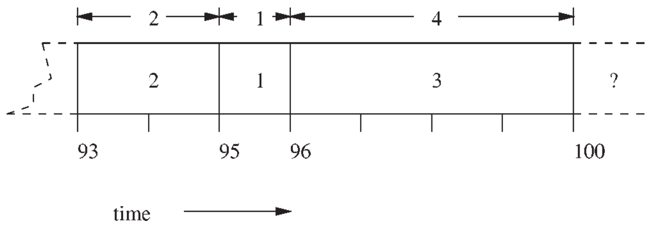


Figure 1. Example 1.

In general, as shown in section 6, the proposed 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 2. Consider the following parameters: $M = 2$, $l_1 = l_2 = 1$, $p_1 = 0.2 + \varepsilon$, $p_2 = 1 - p_1$, and $0 < \varepsilon < 0.05$. In this case, the algorithm produces the cyclic schedule (1,2), i.e., $1, 2, 1, 2, \ldots$, 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 $2.9/3 + 2\varepsilon/3 < 1$. Thus, in this case, the algorithm is not optimal. However, the overall mean access time 1.0 of the algorithm is within 3.5% of that achieved by the cyclic schedule (1,2,2).

A drawback of 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 shown in the next section.

It can be shown that if ties occurring in step 2 of algorithm A are broken deterministically, then the resultant schedule is cyclic [18]. However, determining the cycle itself is compute-intensive. Also, the cycle size is often very large, potentially making it impractical to store the entire schedule. Secondly, producing the schedule at “run-time” has the advantage that any changes in parameters such as demand probabilities can be taken into account.[1] Therefore, techniques to reduce the time complexity are of interest. Also note that the results obtained in relation to the bucketing scheme are also useful to optimize the previously proposed multidisk [1] scheme.

3.2. *Scheduling algorithm with bucketing*

Partition 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 $q_j = (\sum_{i \in B_j} p_i)/m_j$ as the average demand probability of the items in bucket B_j, and $d_j = (\sum_{i \in B_j} l_i)/m_j$ as the average length of the items in bucket B_j. Note that $\sum_{j=1}^{k} m_j q_j = 1$. As shown in appendix B, to minimize the overall mean access time, the following condition must hold when bucketing is used: if item i is in bucket B_j, then spacing $s_i \propto \sqrt{d_j / q_j}$.

In other words,

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

The scheduling algorithm with *bucketing* is based on the above result. Let Q be the current time and $R(i)$ be the

[1] For instance, demand probabilities may change as and when new clients subscribe the broadcast service, or existing clients unsubscribe.

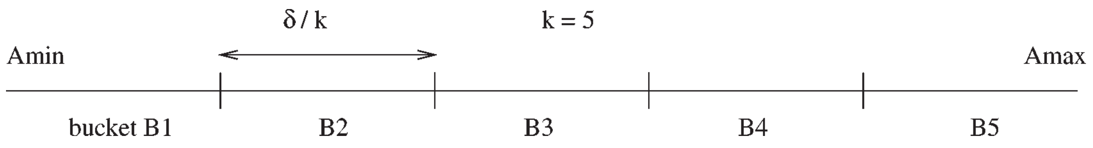


Figure 2. Heuristic for assigning items to k buckets: The interval $(A_{\min}, A_{\max})$ is divided into k equal-sized sub-intervals. An item i whose $\sqrt{p_i/l_i}$ value belongs to the jth sub-interval is assigned to bucket B_j $(1 \leqslant j \leqslant k)$.

time when item i was most recently broadcast. Let I_j denote the item at the front of bucket B_j. Let $G(j)$ now denote $(Q - R(I_j))^2 q_j/d_j$, $1 \leqslant j \leqslant k$. Function $G(j)$ used here is similar (but not identical) to function $G(j)$ used in algorithm A in the previous section. The algorithm with *bucketing*, named algorithm B, is obtained from the above result.

Algorithm B using bucketing

Step 1. Determine maximum value of $G(j) = (Q - R(I_j))^2 q_j/d_j$ over all buckets j, $1 \leqslant j \leqslant k$. 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 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.

As shown in appendix B, a lower bound on achievable *overall mean access time* using bucketing is given by

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

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 the purpose of performance evaluation, 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 values of $\sqrt{p_i/l_i}$ $(1 \leqslant i \leqslant M)$, 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 $(1 \leqslant j \leqslant k)$ if $(j-1)\delta/k < (\sqrt{p_i/l_i} - A_{\min}) \leqslant (j\delta/k)$. This is pictorially depicted in figure 2. The above heuristic executes in O(M) time, and needs to be executed once for given probability and length distributions.

The notion of a *bucket* is similar to that of a *broadcast disk* in the multi-disk approach proposed by Acharya et al. [1]. Therefore, the result in equation (5) can be used to determine suitable frequencies for the broadcast disks. The differences between *buckets* and *broadcast disks* are summarized in section 7.

4. Effect of transmission errors on scheduling strategy

The algorithms presented in section 3 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 possible for the client to ask the server to retransmit the data. If a client waiting for item i receives an instance of item i with *uncorrectable* errors, the item is discarded by the client. The client must wait for the next instance of item i. In this section, we evaluate the impact of uncorrectable errors on the scheduling strategy for broadcasts.

Suppose that uncorrectable errors occur in an item of length l with probability $E(l)$. Now, l_i denotes length of item i after encoding with an error control code. As shown in appendix C, the *overall mean access time*, t, assuming that instances of item i are equally spaced with spacing s_i, is given by

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

To take uncorrectable errors into account, the square-root rule in theorem 1 needs to be modified 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

$$s_i \propto \sqrt{\frac{l_i}{p_i}\frac{1-E(l_i)}{1+E(l_i)}}.$$

The proof of theorem 2 is very similar to that of theorem 1. Note that, when all items have the same length, the term $(1-E(l_i))/(1+E(l_i))$ becomes a constant (independent of i). Therefore, in this case, theorem 2 reduces to theorem 1. The lower bound on the *overall mean access time* now becomes [19]

$$t_{\text{opt\_error}} = \frac{1}{2}\left(\sum_{i=1}^{M}\sqrt{p_i l_i \frac{1+E(l_i)}{1-E(l_i)}}\right)^2. \tag{7}$$

Theorem 2 implies that in an optimal schedule,

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

The 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

$$G(j) = \big(Q - R(j)\big)^2 (p_j/l_j)\frac{1+E(l_j)}{1-E(l_j)}, \quad 1 \leqslant j \leqslant M.$$

Section 6 evaluates the modified algorithm A (using the re-defined function $G(j)$).

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 broadcasts information on multiple channels, and different clients listen to different number of channels depending on the desired quality of service (as characterized by the mean access time). In this section, we present an algorithm for scheduling broadcast on multiple channels such that the *overall mean access time*, averaged over all clients, is minimized.

Figure 3 illustrates multiple channel broadcast assuming that the number of items is 4 and number of channels is 2. In this illustration, each item is assumed to be of length 1. In figure 3(a), if a client listens only to channel 1, the overall mean access time is 2 time units, as each item is transmitted once every 4 time units on channel 1. On the other hand, if a client listens to both the channels simultaneously, then the overall mean access time is 1 (when a client listens to both channels, it receives each item once every 2 time units).

In general, in a multiple channel schedule, all items are transmitted on each channel. However, under certain circumstances this is not necessary. For instance, in the above illustration, assume that *all* clients listen to both channels (i.e., no client listens to only one channel). In this case, the schedule shown in figure 3(b) may be used instead of that in figure 3(a). In this case too, the overall mean access time is 1 time unit (even though only half the items are transmitted on each channel). However, if some clients listen to only a single channel, then the schedule in figure 3(b) would lead to "starvation", and effectively an *infinite* access time for some requests. Therefore, our algorithm does *not* explicitly partition the items across different channels, and each channel may transmit all the items.

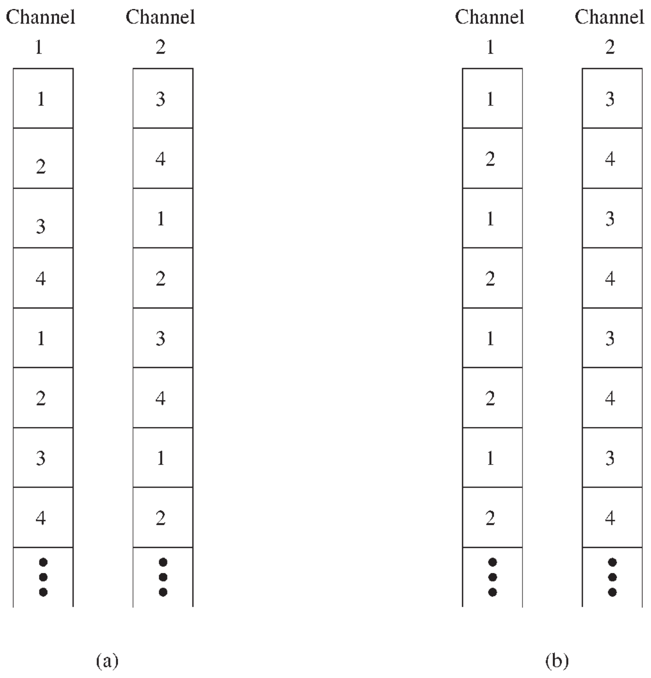


Figure 3. Two examples of multiple channel broadcast schedules.

The approach considered here uses a modification of algorithm A, described in section 3.1, 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 $H = \{1, 2, \ldots, c\}$ denote the set of all broadcast channels. A client 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 $t_{\text{multichan}}$, and obtained as

$$t_{\text{multichan}} = \sum_{S \subseteq H} \Pi_S t_S, \tag{8}$$

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

$t_{\text{multichan}} = \Pi_{\{1\}}t_{\{1\}} + \Pi_{\{2\}}t_{\{2\}} + \Pi_{\{1,2\}}t_{\{1,2\}}$. Equation (3) presented a lower bound (t_{optimal}) on the overall mean access time when a client listens to only one channel. Clearly, for a non-empty set of channels S, a lower bound on t_S is given by $t_{\text{optimal}}/|S|$, where $|S|$ is the number of channels in set S. It follows that a lower bound on $t_{\text{multichan}}$ is given by

$$t_{\text{multichan\_optimal}} = \sum_{S \subseteq H, |S|>0} \Pi_S \frac{t_{\text{optimal}}}{|S|}. \tag{9}$$

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

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

Now we present an algorithm to schedule broadcast on multiple channels. This algorithm is obtained by generalizing algorithm A in section 3.1. In the following, assume that current time is Q, and the algorithm needs to determine which item to broadcast on channel h (where $1 \leqslant h \leqslant c$). Let $R_h(j)$ denote the most recent time[2] when item j was broadcast on channel h ($1 \leqslant h \leqslant c$, $1 \leqslant j \leqslant M$). $R_h(j)$ 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 ($1 \leqslant j \leqslant M$). $G_h(j)$ is defined as follows:

$$G_h(j) = \frac{p_j}{l_j}\left(\sum_{S \subseteq H, h \in S} \Pi_S \big(Q - R^S(j)\big)^2\right). \tag{10}$$

Function $G(j)$ used in algorithm A was motivated by theorem 1. The above definition of function $G_h(j)$ is obtained by generalizing function $G(j)$, by observing the differences between the expressions for overall mean access time for single channel and multiple channel broadcasts (as given in equations (1) and (8)). Note that the summation in the expression for $G_h(j)$ is over all subsets S of H that contain channel h. In particular, when $c = 2$, we have $H = \{1, 2\}$, and

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

and

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

The proposed algorithm is as follows.

Algorithm for channel h, $1 \leqslant h \leqslant c$

Step 1. $R^S(j) = \max_{h \in S} R_h(j)$, $\forall S, \forall j$, $S \subseteq H$, $1 \leqslant j \leqslant M$.

[2] $R_h(j)$ is analogous to $R(j)$ used in algorithms A and B.

Step 2. Determine maximum $G_h(j)$ over all items j, $1 \leqslant j \leqslant M$. 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 6.5 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* (as described in section 3.2).

6. Performance evaluation

In this section, we evaluate various algorithms presented above, assuming that the number of items $M = 1000$. The evaluation is performed by an *analysis* of the broadcast schedule produced by our algorithms. Analytical evaluation provides accurate overall mean access time without having to conduct multiple simulations to obtain small confidence intervals.

For evaluating a broadcast scheduling algorithm for a particular set of parameters, the broadcast schedule is produced for 2,000,000 time units. For a given broadcast schedule, the overall mean access time is calculated analytically – appendix D describes how a given schedule can be evaluated analytically. An alternative to producing such a large schedule would have been to determine the broadcast cycle produced by the algorithm and determine the overall mean access time for the broadcast cycle. However, the time required to determine the broadcast cycle would be very large in many cases.

6.1. Demand probability distribution

We assume that demand probabilities follow the Zipf distribution (similar assumptions are made by other researchers as well [1,4,21]). The Zipf distribution may be expressed as follows:

$$p_i = \frac{(1/i)^\theta}{\sum_{i=1}^{M}(1/i)^\theta}, \quad 1 \leqslant i \leqslant M,$$

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.

6.2. Length distribution

A *length distribution* specifies length l_i of item i. We consider two distributions.

- Increasing length distribution. Consider the following function:

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

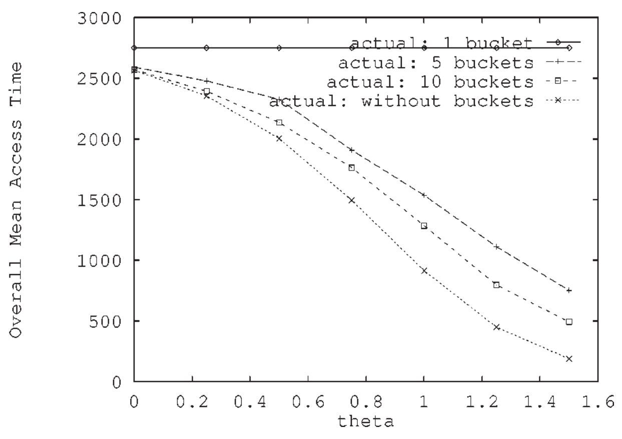
  where L_0 and L_1 are parameters that characterize the distribution. L_0 and L_1 are both positive integers. The `round()` function above returns a rounded integer value of its argument. In this section, we present results for the *Increasing Length Distribution* obtained by assuming $L_0 = 1$ and $L_1 = 10$. Analogous results for a *decreasing* length distribution (with $L_0 = 10$ and $L_1 = 1$) and a *uniform* length distribution (with $L_0 = L_1$) are omitted for brevity [19].
- Random length distribution. In this distribution, we choose integral lengths randomly distributed from 1 to 10 with uniform probability.

6.3. *Performance evaluation in the* absence *of uncorrectable errors*

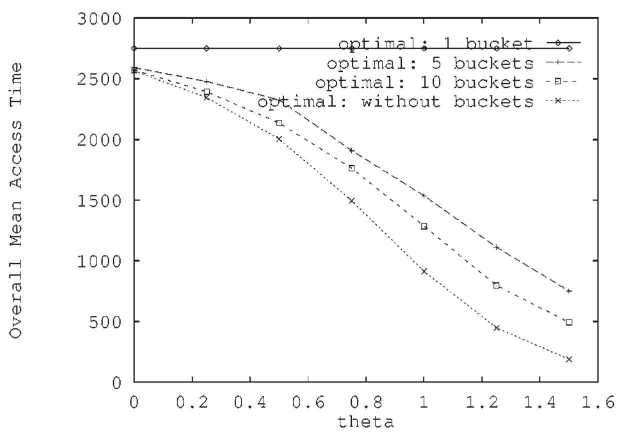
In this section, we evaluate algorithms A and B, assuming that uncorrectable transmission errors do not occur. Performance evaluation in the presence of such errors is discussed in the next section.

Figures 4 and 5 plot *overall mean access time* for different values of access skew coefficient θ, for the two length distributions presented earlier. In each of these figures, the curve titled *without buckets* corresponds to the performance of algorithm A, whereas the curves titled i *buckets* correspond to algorithm B using i buckets. Also, in each figure, part (a) plots the *actual* performance measured for our algorithms, and part (b) plots the "optimal" performance, i.e., the corresponding analytical lower bound on *overall mean access time* (obtained using equations (3) and (6)).

First observation from the performance evaluation results is that the actual performance is very close to the corresponding lower bounds (within less than 1%). Therefore, analytical bounds may be used as an approximation of actual performance. Now note that, when the number of buckets is 1, algorithm B reduces to the so-called "flat" cyclic scheduling [1] 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 than algorithm B, it is interesting to see how the 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

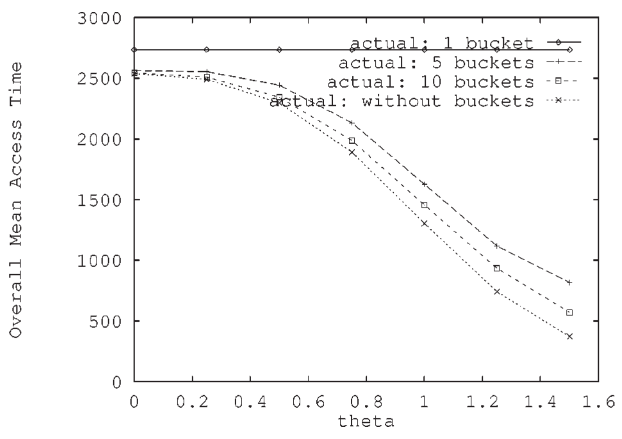


(a) Performance of algorithms A and B

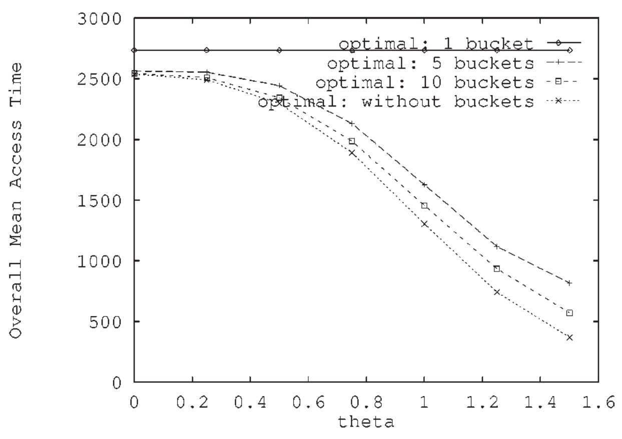


(b) Analytical lower bounds

Figure 4. *Overall mean access time* for different values of access skew coefficient θ and using increasing length distribution.



(a) Performance of algorithms A and B



(b) Analytical lower bounds

Figure 5. *Overall mean access time* for different values of access skew coefficient θ and using random length distribution.

5 buckets is not always adequate to achieve access time comparable with 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), the 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.

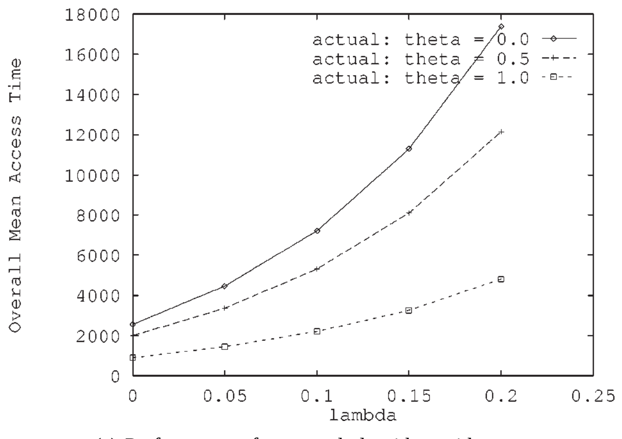
An important conclusion from the above results is that the performance of algorithm B, with a relatively small number of buckets (10 buckets in our illustration), is quite close to that achieved by algorithm A (effectively, using $M = 1000$ buckets). This implies that algorithm B can significantly reduce time complexity, with a reasonably small degradation in performance.

6.4. Performance evaluation in the presence of uncorrectable errors

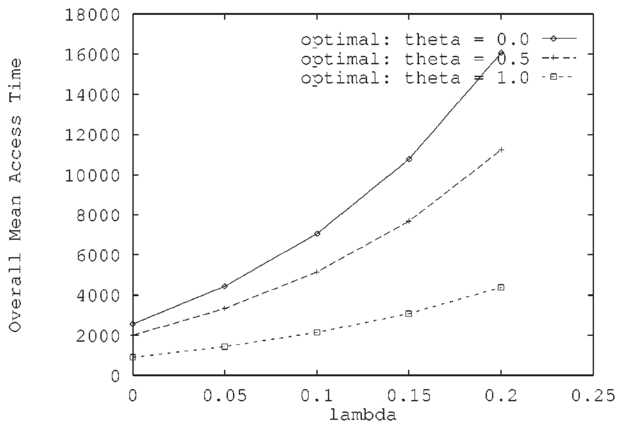
In this section, we evaluate the performance of the algorithm in the presence of uncorrectable errors as explained in section 4. For the sake of illustration, we assume that uncorrectable errors occur according to a Poisson process with rate λ. Hence $E(l_i) = 1 - \mathrm{e}^{-\lambda l_i}$. Figures 6 and 7 plot *overall mean access time* in the presence of errors for different error rates (λ), and for increasing and random length distributions, respectively. In each of these figures, part (a) plots the actual performance obtained using algorithm A modified to take errors into account, and part (b) plots corresponding analytical lower bounds, for $\theta = 0, 0.5$ and 1. The lower bounds are obtained using equation (7) (substituting $E(l_i) = 1 - \mathrm{e}^{-\lambda l_i}$). Note that the results presented in the previous section correspond to the case when $\lambda = 0$. The performance results show that the proposed algorithm A, modified to take errors into account, achieves performance close to optimal (within 3% of optimal for small λ, and within 10% for larger λ). Previous research on broadcasts does not take uncorrectable errors into account when determining the broadcast schedules, or when evaluating the *access time*.

6.5. Performance with multiple broadcast channels

In this section, we evaluate the performance of the algorithm given in section 5 for multiple channel broadcast, assuming the number of channels $c = 2$. We also assume that $\Pi_{\{1\}} = \Pi_{\{2\}} = (1 - \Pi_{\{1,2\}})/2$.

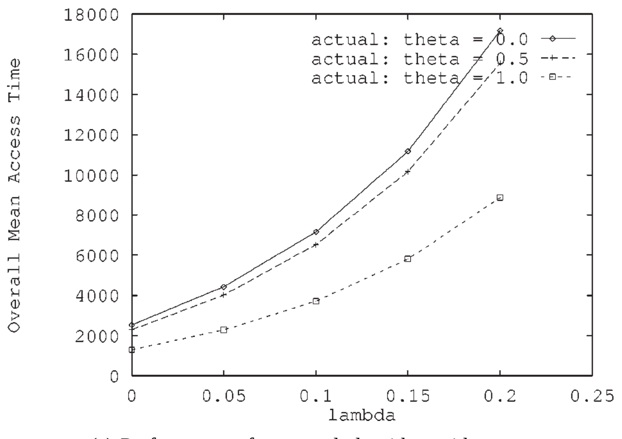


(a) Performance of proposed algorithm with erros

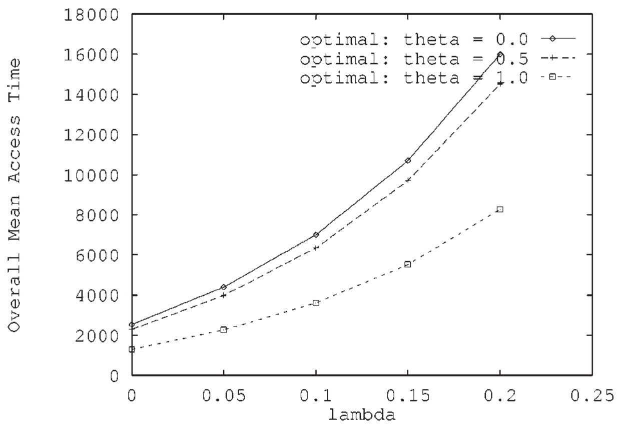


(b) Analytical lower bounds

Figure 6. *Overall mean access time* versus λ for different values of θ and increasing length distribution. The results in figure (a) are obtained using algorithm A modified to take errors into account.

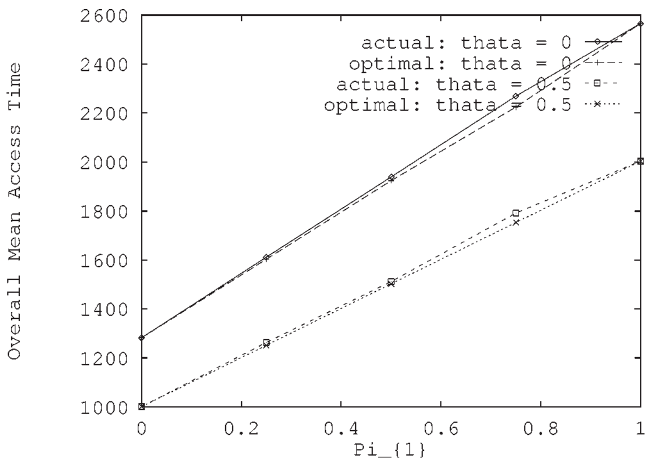


(a) Performance of proposed algorithm with errors

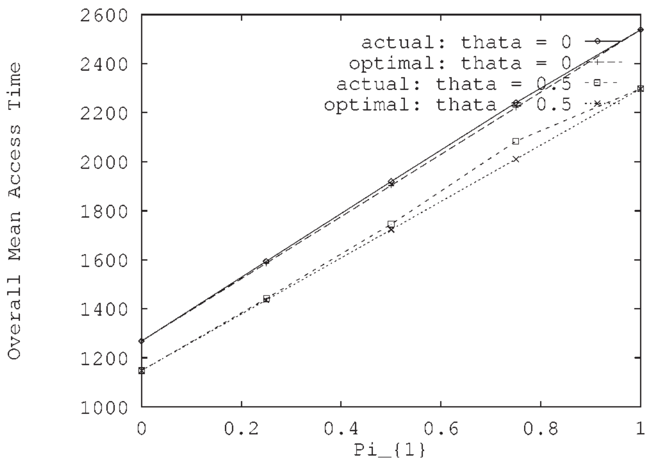


(b) Analytical lower bounds

Figure 7. *Overall mean access time* versus λ for different values of θ and random length distribution. The results in figure (a) are obtained using algorithm A modified to take errors into account.



(a) Increasing length distribution



(b) Random length distribution

Figure 8. Performance evaluation for 2 channels. Label Pi_{1} on the horizontal axis denotes $\Pi_{\{1\}}$. For these graphs, $\Pi_{\{1\}} = \Pi_{\{2\}} = (1 - \Pi_{\{1,2\}})/2$.

Figures 8(a) and (b) show the results for increasing and random length distributions, respectively. Results are plotted for skew coefficient $\theta = 0$ and 0.5, and different values of $\Pi_{\{1\}}$. (In these figures, label Pi_{1} on the horizontal axis denotes $\Pi_{\{1\}}$.) In each figure, the curves labeled *actual* plot the actual performance of our scheduling algorithm, and the curves labeled *optimal* plot the analytical lower bound obtained using equation (9). In all cases, note that the actual performance is very close to optimal (within less than 1% of optimal).

For each algorithm presented in this paper, we have also evaluated their performance using *decreasing* and *uniform* length distributions. The results for these length distributions are similar to those presented here for increasing and random length distributions. For brevity, these additional results are not included here.

7. Related work

Some of the early work relevant to this paper was performed in the context of datacycle [8,12], and teletext and videotex [3,4,11,21] systems. The problem of data broadcasting has received renewed 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* [1,4–6,8,9,15,21] and schemes attempting to reduce the *tuning time* (or power consumption) [10,13,14].

Ammar and Wong [4,21] 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. Algorithm A presented in the paper was obtained using the result in [18, theorem 1]. Su and Tassiulas later independently arrived at this algorithm by a numerical search that concluded that algorithm A is superior in a particular class of scheduling algorithms [17]. An algorithm similar to algorithm A has also been applied to video-on-demand systems [2].

A probabilistic approach for deciding which item to transmit next has been suggested previously [11,20,21]. The probabilistic algorithm was proposed for items of unit length (i.e., $l_i = 1$ for all i). The overall mean access time for the probabilistic algorithm is given by $(\sum_{i=1}^{M} \sqrt{p_i})^2$ (when $l_i = 1$) [21]. On the other hand, algorithm A achieves overall mean access time very close to the lower bound $\frac{1}{2}(\sum_{i=1}^{M} \sqrt{p_i})^2$ (when $l_i = 1$). Thus, the overall mean access time achieved by the proposed algorithm is better than the probabilistic algorithm by approximately a factor of 2.

The bucketing scheme bears some similarities to the multidisk approach proposed by Acharya et al. [1]. The differences between the work in [1] and our work are as follows:

(a) Acharya et al. do not have a way of determining the optimal frequencies for the different disks, whereas our algorithm automatically tries to use the optimal frequencies.

(b) The algorithm in [1] *strictly* 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 [1], is that caching algorithms are simplified under such conditions. However, it is possible to implement caching algorithms similar to those in [1] for the bucketing scheme as well.

(c) Our algorithm works well with items of arbitrary sizes. [1] is constrained to fixed size items. Bar-Noy et al. [7] have recently obtained several interesting theoretical results related to the multidisk approach.

Similar to our discussion in section 4, Jain and Werth [15] 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. ECC is not always sufficient to achieve forward error correction, therefore, uncorrectable errors remain an issue (which is ignored in the past work on data broadcast).

8. Summary

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

- We propose algorithms for scheduling broadcasts, with the goal of minimizing the *access time*. Performance evaluation shows that our algorithms perform quite well (close to the theoretical optimal). The *bucketing* scheme proposed in the paper facilitates a trade-off between time complexity and performance of the scheduling algorithm.
- The paper considers the impact of errors on optimal broadcast schedules. An algorithm for broadcast scheduling in the presence of errors is proposed.
- 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 paper presents an algorithm for scheduling broadcasts in such a system.

Future work will include design of strategies for caching and updates that attempt to achieve optimal performance while incurring low overhead.

Acknowledgements

Authors thank the referees for their comments. Research reported is supported in part by Texas Advanced Technology Program grant 009741-052-C and US National Science Foundation grant MIP-9423735. Preliminary version of this paper appeared at the 1996 Workshop on Satellite Based Information Services, Rye, NY.

Appendix A. Proof of theorem 1

Let $l_i/s_i = r_i$. Note that, with the equal-spacing assumption, r_i is the fraction of bandwidth allocated to item i. Therefore, $\sum_{i=1}^{M} r_i = 1$. Also, note that $s_i = l_i/r_i$. Therefore, equation (2) can be rewritten as

$$t = \frac{1}{2}\sum_{i=1}^{M}\frac{p_i l_i}{r_i}. \tag{11}$$

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 $\partial t/\partial r_i = 0$, $\forall i$. We now solve these equations, beginning with $0 = \partial t/\partial r_1$.

$$\begin{aligned}
0 = \frac{\partial t}{\partial r_1} &= \frac{1}{2}\frac{\partial}{\partial r_1}\left(\sum_{i=1}^{M}\frac{p_i l_i}{r_i}\right)\\
&= \frac{1}{2}\frac{\partial}{\partial r_1}\left(\frac{p_1 l_1}{r_1} + \sum_{i=2}^{M-1}\frac{p_i l_i}{r_i} + \frac{p_M l_M}{\left(1-\sum_{i=1}^{M-1} r_i\right)}\right)\\
&= \frac{1}{2}\left(-\frac{p_1 l_1}{r_1^2} + \frac{p_M l_M}{\left(1-\sum_{i=1}^{M-1} r_i\right)^2}\right)\\
\Rightarrow \frac{p_1 l_1}{r_1^2} &= \frac{p_M l_M}{(1-\sum_{i=1}^{M-1} r_i)^2}.
\end{aligned} \tag{12}$$

Similarly,

$$\frac{p_2 l_2}{r_2^2} = \frac{p_M l_M}{\left(1-\sum_{i=1}^{M-1} r_i\right)^2}. \tag{13}$$

From equations (12) and (13), we get

$$\frac{p_1 l_1}{r_1^2} = \frac{p_2 l_2}{r_2^2} \Rightarrow \frac{r_1}{r_2} = \sqrt{\frac{p_1 l_1}{p_2 l_2}}.$$

Similarly it can be shown that

$$\frac{r_i}{r_j} = \sqrt{\frac{p_i l_i}{p_j l_j}}, \quad \forall i, j.$$

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 $a = 1/\sum_{j=1}^{M}\sqrt{p_j l_j}$ exists such that $r_i = a\sqrt{p_i l_i}$ is the *only* possible solution for the equations $\partial t/\partial r_i = 0$, such that $\sum_{i=1}^{M} r_i = 1$. 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 $r_i = \sqrt{p_i l_i}/\sum_{j=1}^{M}\sqrt{p_j l_j}$ into equation (11), and simplifying, yields optimal *overall mean access time* as $t = \frac{1}{2}(\sum_{i=1}^{M}\sqrt{p_i l_i})^2$.

Appendix B. Bucketing scheme

This section presents a derivation of equation (6). Bucket B_j ($1 \leqslant j \leqslant k$) contains m_j items, such that $\sum_{j=1}^{k} m_j = M$. Also, $q_j = (\sum_{i\varepsilon B_j} p_i)/m_j$ and $d_j = (\sum_{i\varepsilon B_j} l_i)/m_j$ are average demand probability and average length of the items in bucket B_j, respectively. The proof here is similar to the proof in appendix A. For optimal solution, the items should be equally spaced. With bucketing, the spacing for all items in the same bucket is also identical. We define S_j as the spacing between consecutive instances of an item in bucket B_j.

Let T_j denote the *item mean access time* of an item in bucket B_j. Then, $T_j = \frac{1}{2}S_j$. Note that, with the equal spacing assumption, *item mean access time* is identical for all items in the same bucket. Thus, the *Overall Mean Access Time* is given by

$$t = \sum_{j=1}^{k}\left(\sum_{i\in B_j} p_i\right)T_j = \sum_{j=1}^{k}\left(\sum_{i\in B_j} p_i\right)\frac{S_j}{2}.$$

Since $\sum_{i\in B_j} p_i = m_j q_j$, the above equation can be written as $t = \sum_{j=1}^{k}(q_j m_j S_j/2)$. Now, let r_j denote the fraction of bandwidth used for transmitting items from bucket B_j. Thus, $\sum_{j=1}^{k} r_j = 1$. Also, it follows that $r_j = m_j d_j/S_j$. In other words, $S_j = m_j d_j/r_j$. Substituting this expression for S_j, the previous expression for t can now be rewritten as

$$t = \frac{1}{2}\sum_{j=1}^{k}\frac{q_j m_j^2 d_j}{r_j}. \qquad (14)$$

If we denote $q_j m_j^2 d_j$ as X_j, the above equation becomes $t = \frac{1}{2}\sum_{j=1}^{k}(X_j/r_j)$, where $\sum_{j=1}^{k} r_j = 1$. This equation has the same form as equation (11). Therefore, from the proof in appendix A it follows that, with bucketing, to minimize t the following condition must be true:

$$r_j \propto \sqrt{X_j}. \qquad (15)$$

As $\sum_{j=1}^{k} r_j = 1$, $r_j = \sqrt{X_j}/\sum_{i=1}^{M}\sqrt{X_j}$. Substituting this into equation (14), 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 (15), we get

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

As $r_j = m_j d_j/S_j$, we get $S_j \propto \sqrt{d_j/q_j}$. Finally, note that, for each item i in bucket B_j, item spacing s_i is equal to S_j.

Appendix C. Overall mean access time in the presence of errors

Consider item i, instances of which are equally spaced s_i time units apart. Recall 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 $E(l_i)/(1 - E(l_i))$. Thus, the *item mean access time* is obtained as

$$\begin{aligned} t_i &= \frac{s_i}{2} + s_i\frac{E(l_i)}{1-E(l_i)} = s_i\frac{1}{2} + \frac{E(l_i)}{1-E(l_i)} \\ &= \frac{1}{2}s_i\frac{1+E(l_i)}{1-E(l_i)}. \end{aligned}$$

Therefore,

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

Appendix D. Analytical evaluation of a broadcast schedule

In the analytical evaluation, the *item mean access time* t_i for each item i is calculated independently. Then, these are used to obtain the overall mean access time. The results obtained by analytical evaluation are similar to those obtained by simulations.

Let S_{ij} denote the spacing between jth and $(j+1)$th instances of item i in the broadcast schedule. Let there be a total of n instances of item i in the broadcast schedule under evaluation. Note that the number of instances n may be different for different items i. Figure 9 shows the instances of item 1 in a broadcast schedule. In this example, the schedule contains 4 instances of item 1 (note that we produce a finite size schedule for analysis). The spacings S_{11}, S_{12} and S_{13} are shown in the figure. Note that, although our algorithm tries to maintain a constant spacing, in general, S_{ia} may not be equal to S_{ib}, when $a \neq b$.

Let μ be the rate of request arrival. Then, $p_i\mu$ is the rate of arrival of requests for item i. Then, the expected number of requests for item i that arrive between its jth and $(j+1)$th instances in the broadcast is $p_i\mu S_{ij}$. These requests, on average, encounter access time of $S_{ij}/2$. Thus, the mean access time for all requests for item i that arrive between the first and nth instances of item i in the broadcast schedule is given by

$$\frac{\sum_{j=1}^{n-1}(p_i\mu S_{ij})(S_{ij}/2)}{\sum_{j=1}^{n-1}(p_i\mu S_{ij})}.$$

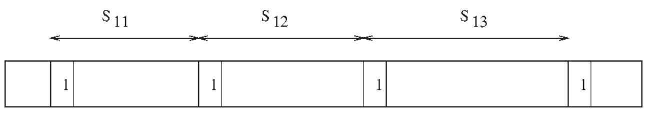


Figure 9. Instances of item i.

We use the value obtained above as the estimate of item mean access time t_i. After simplifying, the above expression becomes

$$t_i = \frac{\sum_{j=1}^{n-1} p_i S_{ij}^2/2}{\sum_{j=1}^{n-1} p_i S_{ij}}.$$

Thus, t_i is not dependent on the request arrival rate μ. Now, using the above estimate of t_i, the overall mean access time is estimated as

$$\sum_{i=1}^{M} p_i t_i.$$

Note that, in the above derivation, we consider the duration between the first and the last instance of each item i to obtain the estimate of t_i. Therefore, the actual begin time of the schedule used for calculating t_i is different for different i. This may introduce a small error in our estimate of t_i, however, the error is negligible when the chosen schedule size is large.

References

[1] S. Acharya, M. Franklin and S. Zdonik, Dissemination-based data delivery using broadcast disks, IEEE Personal Communication (December 1995) 50–60.

[2] C.C. Aggarwal, J.L. Wolf and P.S. Yu, The maximum factor queue length batching scheme for video-on-demand systems, Technical Report RC 20621, IBM T.J. Watson Research Center (November 1996).

[3] M.H. Ammar and J.W. Wong, The design of teletext broadcast cycles, Performance Evaluation 5 (November 1985) 235–242.

[4] M.H. Ammar and J.W. Wong, On the optimality of cyclic transmission in teletext systems, IEEE Transactions on Communications (January 1987) 68–73.

[5] S. Banerjee and V.O.K. Lee, Evaluating the distributed datacycle scheme for a high performance distributed system, Journal of Computing and Information 1 (May 1994).

[6] S. Banerjee, V.O.K. Lee and C. Wang, Distributed database systems in high-speed wide-area networks, IEEE Journal on Selected Areas in Communications 11 (May 1993) 617–630.

[7] A. Bar-Noy, R. Bhatia, J. Naor and B. Schieber, Minimizing service and operation costs of periodic scheduling, Technical Report, Tel Aviv University (1997).

[8] T.F. Bowen et al., The datacycle architecture, Communications of ACM 35 (December 1992) 71–81.

[9] T. Chiueh, Scheduling for broadcast-based file systems, in: *MOBIDATA Workshop* (November 1994).

[10] A. Datta, A. Celik, J.G. Kim, D.E. VanderMeer and V. Kumar, Adaptive broadcast protocols to support efficient and energy conserving retrieval from databases in mobile computing environments, in: *Data Engineering Conference* (April 1997).

[11] J. Gescei, *The Architecture of Videotex Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1983).

[12] G. Herman, G. Gopal, K.C. Lee and A. Weinrib, The datacycle architecture for very high throughput, in: *Proc. of ACM SIGMOD* (1987).

[13] T. Imielinski, S. Viswanathan and B.R. Badrinath, Energy efficient indexing on air, in: *International Conference on Management of Data* (May 1994) pp. 25–36.

[14] T. Imielinski, S. Viswanathan and B.R. Badrinath, Data on the air – organization and access, IEEE Transactions of Data and Knowledge Engineering (July 1996).

[15] R. Jain and J. Werth, Airdisks and airraid: Modelling and scheduling periodic wireless data broadcast (extended abstract), DIMACS Technical Report 95-11, Rutgers University (May 1995).

[16] S. Jiang and N.H. Vaidya, Scheduling algorithms for a data broadcast system: Minimizing variance of the response time, Technical Report 98-005, Computer Science Department, Texas A&M University, College Station (February 1998).

[17] C.J. Su and L. Tassiulas, Broadcast scheduling for the distribution of information items of unequal length, Technical Report, Institute for Systems Research, University of Maryland (1997).

[18] N.H. Vaidya and S. Hameed, Data broadcast scheduling: On-line and off-line algorithms, Technical Report 96-017, Computer Science Department, Texas A&M University, College Station (July 1996).

[19] N.H. Vaidya and S. Hameed, Scheduling data broadcast in asymmetric communication environments, Technical Report 96-022, Computer Science Department, Texas A&M University, College Station (November 1996).

[20] S. Viswanathan, Publishing in wireless and wireline environments, Ph.D. thesis, Rutgers (November 1994).

[21] J.W. Wong, Broadcast delivery, in: *Proceedings of IEEE* (December 1988) pp. 1566–1577.

Nitin Vaidya received the Ph.D. degree from the University of Massachusetts at Amherst in 1992. He previously received M.E. and B.E. (Hons) degrees from the Indian Institute of Science, Bangalore, and the Birla Institute of Technology and Science, Pilani, respectively. He is currently an Associate Professor of Computer Science at the Texas A&M University. His research interests include networking, mobile computing and fault-tolerant computing. Nitin Vaidya is a recipient of a 1995 CAREER award from the National Science Foundation. He has served on program and organizing committees of several conferences. He is a member of the ACM and the IEEE Computer Society.
E-mail: vaidya@cs.tamu.edu

Sohail Hameed received a B.S. in computer engineering from NED University of Engineering and Technology, Pakistan, and an M.S. in computer science from Texas A&M University at College Station. He is currently a systems engineer at Compaq Computer Corporation. His research interests include broadcast scheduling and high-reliability mass-storage system design. He is an associate member of the IEEE.