Efficient Algorithms for Scheduling Single and Multiple Channel Data Broadcast*

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

Technical Report 97-002

February 11, 1997

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. It has been proposed that in such systems the server should broadcast the information periodically. A *broadcast schedule* determines what is broadcast by the server and when.

In this report, we present an algorithm for scheduling broadcast in such environments. This algorithm is based on a *fair queueing* algorithm [6], and can be executed in $O(\log M)$ time, where M is the number of information items. The algorithm significantly improves the time-complexity over previously proposed broadcast scheduling algorithms. We evaluate performance of the algorithm and find it to be close to optimal. We also present an algorithm to coordinate broadcasts over multiple channels, and evaluate its performance for two channels.

Key Words: Data broadcast, asymmetric communication environments, broadcast scheduling, simulation results.

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

Contents

Preface: This report discusses scheduling algorithms for single channel and multiple channels. This problem is also addressed in [23]. This report, however, gives a new algorithm for multiple channel scheduling which is not only more general but also more efficient in terms of performance than [23]. However, the single channel scheduling algorithm proposed in [23] is again presented here for the sake of continuity. This report does not address transmission errors unlike [23]. However, the algorithms presented here can easily be modified to take transmission errors into account.

1 Introduction

Mobile computing and wireless networks are fast-growing technologies that are making ubiquitous computing a reality. Mobile and wireless computing systems have found many applications, including Defense Messaging System (DMS), Digital Battlefield and Data Dissemination (BADD) [9], and as a general-purpose computing tool. With the increasing popularity of portable wireless computers, mechanisms to efficiently transmit information to such *clients* are of significant interest [9]. For instance, such mechanisms could be used by a *satellite* [21] or a *base station* [2] to communicate information of common interest to wireless hosts. In the environment under consideration, the *downstream* communication capacity, from server to clients, is relatively much greater than the *upstream* communication capacity, from clients to server. Such environments are, hence, called *asymmetric* communication environments [2]. In an asymmetric environment, *broadcasting* the information is an effective way of making the information available simultaneously to a large number of users. For asymmetric environment, researchers have previously proposed algorithms for designing *broadcast schedules* [2, 3, 4, 8, 10, 13, 14, 15, 16, 11, 20, 21, 25, 26].

We consider a database that is divided into *information items*. The server periodically broadcasts these items to all clients. A broadcast *schedule* determines when each item is transmitted by the server. We present a new approach to design broadcast schedules that attempts 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 [2, 3, 4, 8, 10, 15, 16, 25, 26].

The algorithms presented in this report are *on-line* algorithms. An on-line algorithm does *not* a priori generate the broadcast schedule. Instead, the algorithm determines which item to broadcast next when the server is ready to broadcast an item. On-line algorithms are of interest as they can quickly adapt to time-varying environments.

The time-complexity involved in determining the next item to broadcast is critical. Our previous work [21] includes a number of on-line algorithms each with linear time complexity in number of items. Linear time-complexity in number of items may become intolerable when the server has a large number of items to broadcast. Techniques like bucketing have previously been used to reduce the complexity while sometimes compromising performance [21]. This report presents two algorithms, based on *packet fair queueing* [6, 5, 18, 19], both having the time-complexity of $O(\log M)$, where M is the number of information items. This is a significant improvement in time-complexity over previously proposed algorithms with comparable performance. The proposed algorithms can easily be extended to take transmission errors into account as wireless environments are subject to such errors [23].

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. We extend the proposed algorithm to a system where the server can broadcast simultaneously on multiple channels, and the clients may listen to one channel or both channels.

The rest of the report is organized as follows. Section 2 introduces terminology, and derives some theoretical results that motivate the proposed algorithms. Section 3 presents proposed scheduling algorithm for single channel. This algorithm was also presented in [23]. However, the performance evaluation of the algorithm is done using different parameters in this report. Section 4 presents scheduling algorithm for broadcast on multiple channels. The algorithm presented here is completely different from the algorithm presented in [23]. Besides, the multiple channel algorithm in [23] gives scheme to schedule only on two channels and is hard to generalize. Whereas, the Multiple Channel Scheduling Algorithm presented in Section 4 of this report can be used to schedule any number of channels and performs better than the previously proposed algorithm. Section 5 evaluates the performance of our algorithms. Related work is discussed in Section 6. A summary is presented in Section 7.

2 Theoretical Foundation for the Proposed Algorithms [22]

First we introduce some terminology and notations to be used here [22].

- Database at the server is assumed to be divided into many *information items*. The items are <u>not</u> necessarily of the same length.
- l_i represents length of item i.
- 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 = total number of information items in the server's database. The items are numbered 1 through M.

- The broadcast consists of a cycle of size N time units. (For an acyclic schedule, $N = \infty$.) The broadcast schedule is repeated after N time units (if N is finite).

  Figure 1 illustrates broadcast cycle (1,2,1,3). That is, the items transmitted by the server are $1, 2, 1, 3, 1, 2, 1, 3, 1, 2, 1, 3, \cdots$. Assume that $l_1 = 1$, $l_2 = 2$ and $l_3 = 3$. Then, size of the broadcast cycle (1,2,1,3) is $N = l_1 + l_2 + l_1 + l_3 = 1 + 2 + 1 + 3 = 7$.

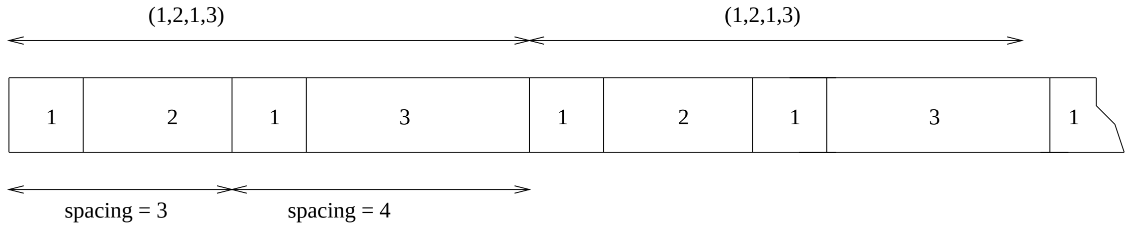


Figure 1: *Broadcast schedule*

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

- *Frequency of an item* : *Frequency* f_i of item i is the number of instances of item i in the broadcast cycle. The f_i instances of an item are numbered 1 through f_i. Size of the broadcast cycle is given by $N = \sum_{i=1}^{M} f_i l_i$, where l_i is the length of item i. In the cycle (1,2,1,3) in Figure 1, $f_1 = 2$ and $f_2 = f_3 = 1$.

- *Spacing* : The *spacing* between two instances of an item is the time it takes to broadcast information from the beginning of the first instance to the beginning of the second instance. s_{ij} denotes the spacing between j-th instance of item i and the next instance of item i ($1 \leq j \leq f_i$). Note that, after the f_i-th instance of an item in a transmission of the broadcast cycle, the next instance of the same item is the *first* instance in the next transmission of the broadcast cycle. For example, in Figure 1, $s_{11} = 3$ and $s_{12} = 4$.

  If all instances of an item i are equally spaced, then s_i denotes the spacing for item i. That is, $s_{ij} = s_i$, $1 \leq j \leq f_i$.

- *Item Mean Access Time* of item i, denoted t_i, is defined as the average wait by a client needing item i until it starts receiving item i from the server. It can be shown that the *item mean access time* is minimized when all instances of the item are equally spaced. That is, $s_{ij} = s_i$ for all j [16]. Hereafter, for our theoretical development, we assume that all instances of item i are spaced s_i apart. This assumption cannot always be realized in practice (See Appendix B), however, the assumption does provide a basis for developing the proposed algorithms.

We assume that a client is equally likely to need an item at any instant of time (uniform distribution). Then, the *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. Hence,

$$t_i = \frac{s_i}{2} \tag{1}$$

- *Demand probability* : *Demand probability* p_i denotes the probability that an item needed by a client is item i. The demand probability distribution affects the optimal broadcast schedule. As intuition suggests, items with greater demand probability should be broadcast more frequently than items with smaller demand probability. We will later determine the optimal broadcast frequencies as a function of demand probabilities and other parameters.

- *Overall Mean Access Time*, denoted $t_{overall}$, is defined as the average wait encountered by a client (averaged over all items). Thus,

$$t_{overall} = \sum_{i=1}^{M} t_i \, p_i$$

  Using Equation 1, we obtain $t_{overall}$ as

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

The theorem below provides a theoretical basis for the proposed scheduling scheme.

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

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

Proof: Appendix A presents the proof. □

From Theorem 1 it follows that, there exists a constant K such that $f_i = K\sqrt{\frac{p_i}{l_i}}$. Now note that, cycle size $N = \sum_{i=1}^{M} f_i l_i$. Substituting the expression for f_i into this equality, and solving for K, yields

$$K = \frac{N}{\sum_{i=1}^{M} \sqrt{p_i \, l_i}}$$

As spacing $s_i = N/f_i$, for *overall mean access time* to be minimized, we need

$$\begin{aligned} s_i &= \frac{N}{K}\sqrt{\frac{l_i}{p_i}} \\ &= \left(\sum_{j=1}^{M}\sqrt{p_j l_j}\right)\sqrt{\frac{l_i}{p_i}} \end{aligned} \tag{3}$$

Substituting this expression for s_i into Equation 2, the optimal *overall mean access time*, named $t_{optimal}$, is obtained as:

$$t_{optimal} = \frac{1}{2}\left(\sum_{i=1}^{M}\sqrt{p_i\, l_i}\right)^2 \tag{4}$$

(Appendix A also presents a derivation of the above expression.)

$t_{optimal}$ is derived assuming that instances of each item are equally spaced. As illustrated in Appendix B, the *equal-spacing* assumption cannot always be realized. Therefore, $t_{optimal}$ 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.

3 Proposed Broadcast Scheduling Scheme [23]

In this section, we consider the case when the information items are broadcast on a single channel. Section 4 considers multiple channel broadcast.

As noted above, for an optimal schedule, spacing between consecutive instances of item i should be obtained using Equation 3. Equation 3 can be rewritten as

$$\frac{l_i}{s_i} = \frac{l_i}{\left(\sum_{j=1}^{M}\sqrt{p_j l_j}\right)\sqrt{\frac{l_i}{p_i}}} \tag{5}$$

Let ϕ_i denote the right-hand side of Equation 5. That is, $\phi_i = \frac{l_i}{\left(\sum_{j=1}^{M}\sqrt{p_j l_j}\right)\sqrt{\frac{l_i}{p_i}}}$. Then, we have $l_i/s_i = \phi_i$. Thus, the two conditions for obtaining an optimal schedule are: (i) $\frac{l_i}{s_i} = \phi_i$ for each item i, and (ii) all instances of each item i should be spaced equally apart with spacing s_i. Note that $\frac{l_i}{s_i}$ is the *fraction of broadcast bandwidth* allocated to item i. It turns out that the above two conditions are similar to those imposed on "packet fair queueing" algorithms [5, 6]. Although the problem of *packet fair queueing* is not identical to broadcast scheduling, the similarities between these two problems motivated us to adapt a packet fair queueing algorithm in [5, 6] to broadcast scheduling. The broadcast scheduling algorithm, thus obtained, is presented below.

For each item i, the algorithm maintains two variables, B_i and C_i. B_i is the earliest time when next instance of item i should *begin* transmission, and $C_i = B_i + s_i$. (It may help the reader to interpret C_i as the "suggested worst-case *completion* time" for the next transmission of item i.)

Single Channel Broadcast Scheduling Algorithm

Step 0: Determine optimal spacing s_i for each item i, using Equation 3.
Current time is denoted by T. Initially, $T = 0$.
Initialize $B_i = 0$ and $C_i = s_i$ for $1 \leq i \leq M$.
Step 1: Determine set S of items for which $B_i \leq T$.
That is, $S = \{i \mid B_i \leq T,\ 1 \leq i \leq M\}$.
Step 2: Let C_{min} denote the minimum value of C_i over all items i in set S.
Step 3: Choose item $j \in S$ such that $C_j = C_{min}$. If this equality
holds for more than one item, choose any one of them arbitrarily.
Step 4: Broadcast item j at time T.
$B_j = C_j$
$C_j = B_j + s_j$
Step 5: When item j completes transmission, increment T by l_j.
Go to step 1.

The algorithm iterates steps 1 through 5 repeatedly, broadcasting one item per iteration. In each iteration, first the set S of items with begin times B_i smaller than or equal to T is determined. The items in set S are "ready" for transmission. From among these items, the items with the smallest C_i (suggested worst-case completion time) is chosen for broadcast.

Using the *heap* data structure [12], steps 1 through 4 can be implemented such that, the average time complexity per iteration is $O(\log M)$. Bennett and Zhou [6] cite a $O(\log M)$ fair queueing implementation that can be used to implement the above algorithm. Their implementation is apparently presented in [5]; however, we are unable to obtain a copy of [5] at this time. It is possible that their implementation of fair queueing is analogous to the implementation summarized below. Keshav [17] also presents a heap-based implementation of fair queueing. However, his fair queueing algorithm is somewhat different from that in [6].

We maintain two binary heaps, H_B and H_C. Heap H_B has item with smallest B_i value, among all its items, at its root. H_C has item with smallest C_i value, among all its items, at its root. (Heap H_C implements set S.) Every item belongs to exactly one of the two heaps at any given time. In the beginning, H_B contains all the items and H_C is empty. In Step 1 of the above algorithm, set S can be determined by repeatedly removing items j

from the root of H_B until $B_j > T$ or H_B becomes empty, and inserting them into H_C. Note that after every removal of an item, H_B is to be reheaped. Both insertion and removal of an item in a binary heap (including reheaping) takes $O(\log M)$ time. Step 2 can be performed by removing the root item from H_C again in $O(\log M)$ time. An item j that is broadcast (after removal from H_C) is inserted back into H_B in step 4 (after the new B_j and C_j values are calculated). The insertion requires $O(\log M)$ time as well. Note that, in some iterations, more than one item may be removed from heap H_B (in step 1) and added to heap H_C, while in some iterations no item may be removed from H_B.

Each broadcast instance of an item j is first inserted in H_B, then removed from H_B and inserted into H_C, then removed from H_C, and transmitted. Thus, each item transmitted requires 4 heap operations, resulting in an average time complexity $O(\log M)$. (Another way to arrive at this conclusion is to observe that, because one item is added to heap H_B in each iteration, on average only 1 item can be removed from H_B per iteration.)

As an illustration, assume that the database consists of 3 items, such that $l_1 = 1$, $l_2 = 2$, $l_3 = 3$, $p_1 = 0.5$, $p_2 = 0.25$, and $p_3 = 0.25$. In this case, $s_1 = 3.224$, $s_2 = 6.448$ and $s_3 = 7.989$. In the first iteration of the above algorithm, at step 2, $B_1 = B_2 = B_3 = T = 0$, and $C_1 = 3.224$, $C_2 = 6.448$ and $C_3 = 7.989$. During the first iteration, $S = \{1, 2, 3\}$, as $T = 0$ and for all items $B_i = 0$. As C_1 is the smallest, item 1 is the first item transmitted. During the second iteration of the algorithm, $T = 1$, $B_1 = 3.224$, $B_2 = B_3 = 0$, $C_1 = 6.448$, $C_2 = 6.448$ and $C_3 = 7.898$. Now, $S = \{2, 3\}$ (as $B_2 = B_3 = 0 < T = 1$, and $B_1 > T$). As $C_2 < C_3$, item 2 is transmitted next. Figure 2 shows the first few items transmitted using the above algorithm. After an initial transient phase, the schedule became cyclic with the cycle being (1,2,1,3).

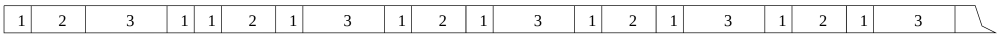


Figure 2: *Illustration of the Single Channel Scheduling Algorithm.*

Simulations show that the above algorithm attempts to use optimal spacing and frequency for each item (i.e., actual spacings and frequencies are approximately equal to the optimal values). Performance measurements for the above algorithm are presented in Section 5. In general, as illustrated in Section 5, the proposed on-line algorithm performs close to the optimal obtained by Equation 4.

4 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 [24], 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 on-line algorithm for scheduling broadcast on multiple channels. Let c denote the total number of channels available to server. We assume that the channels are of equal capacities. A client can listen to any number of channels it wants (can afford). The idea is fairly simple. The server generates a schedule assuming only one channel using Single Channel Scheduling Algorithm explained in Section 3. It then broadcasts successive instances of an item in the single channel schedule on successive channels in a cyclic manner.

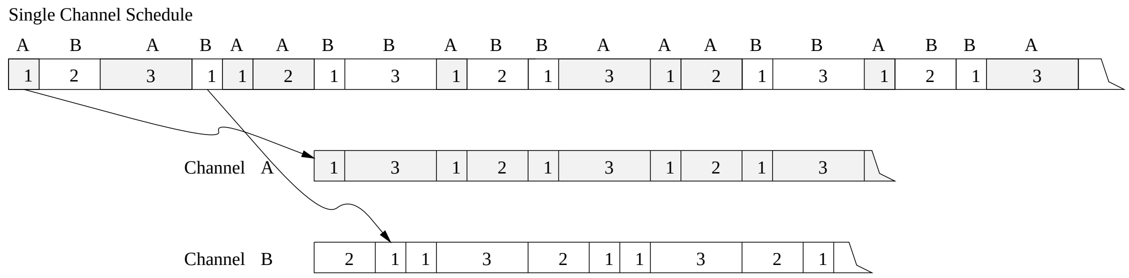


Figure 3: *Illustration of Multiple Channel Scheduling Algorithm for two channels and three items. The server alternately assigns instances of an item from the single channel schedule to the two channels.*

As an example, refer to Figure 3 which illustrates the algorithm for three items: 1, 2 and 3, and for two channels namely A and B. First of all, the server generates a schedule using Single Channel Scheduling Algorithm (Section 3) as shown in Figure 3. (This algorithm can be implemented on-line as shown later.) It then labels the successive instances of item 1 in the schedule as A and B alternately. Similarly successive instances of each of items 2 and item 3 are also labelled as A and B alternately. Note that the labelling for item 1 is started with A, for item 2 with B and for item 3 with C. This is just a heuristic for initialization which most of the time improves the multi-channel schedules. The schedules for channels A and B are then generated by selecting items from single channel schedule sequentially and inserting them in the appropriate channel schedule depending upon their labels. The resulting schedules are shown in Figure 3. Note that most of the time, two different items are broadcast on channels A and B at any instant. This improves the *overall mean access time* for a client listening to both the channels as compared to a client listening to only one of the two channels.

Similarly, Figure 4 shows how the algorithm can be used to generate schedules for three channels namely A, B and C.

The examples above might make the reader believe that the algorithm for scheduling multiple channel broadcast is off-line, that is, the server needs to generate the schedules *a priori*, and that the algorithm cannot be used to determine next item to broadcast on some

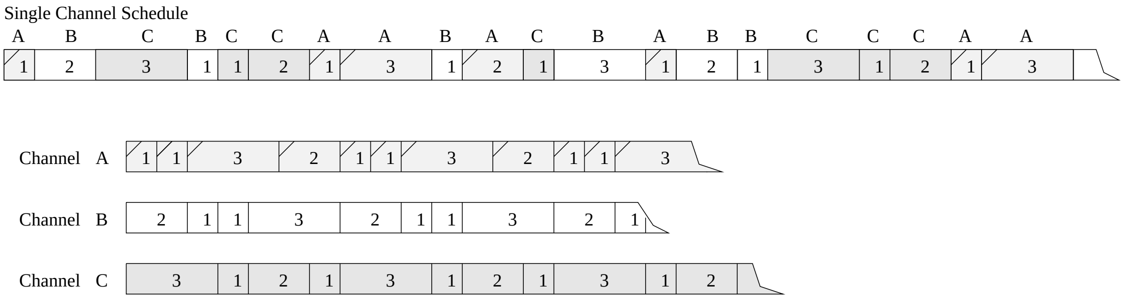


Figure 4: *Illustration of Multiple Channel Scheduling Algorithm for three channels and three items. The successive instances of each of the items from single channel schedule are assigned to the three channels in cyclic fashion.*

particular channel h whenever the channel is idle. However, the same algorithm can easily be made on-line by maintaining queues Q_h for every channel h, $1 \leq h \leq c$ and inserting the items which are scheduled for channel h, if the channel is busy, in queue Q_h. A formal description of algorithm is given hereunder :

Multiple Channel Scheduling Algorithm

Step 0: Determine optimal spacing s_i for each item i, using Equation 3.
Let T denote *virtual time*. (T is just a variable whose value would be used by Single Channel Scheduling Algorithm as current time and its value does not represent actual time in this algorithm. Hence, here T is said to be *virtual* time.) Initialize $T = 0, B_i = 0, C_i = s_i, 1 \leq i \leq M$, and create empty queues Q_h, $1 \leq h \leq c$. Also initialize $last_i = (i \bmod c) + 1$, $1 \leq i \leq M$. $last_i$ holds the channel number over which item i was last broadcast.

The remaining steps are executed to find an item to broadcast next on channel h at any time, $1 \leq h \leq c$.

Step 1: if Q_h is not empty then
{
Step 2: Select item j from the front of Q_h
Step 3: $last_j = h$
}
else
{
Step 4: Use Single Channel Scheduling Algorithm to determine item j to broadcast next using virtual time T, B_i and $C_i, 1 \leq i \leq M$. (The values of T, B_i and C_i are changed by single channel scheduling algorithm as shown in Section 3)

Step 5: Let $nextch = (last_i \bmod c) + 1$
Step 6: if $nextch \neq h$
{
Step 7: enqueue item j at the end of Q_{nextch}
Step 8: $last_j = nextch$
Step 9: goto Step 4
}
Step 10: $last_j = nextch$
}
Step 11: Broadcast item j on channel h

This algorithm, on average, requires $O(\log M)$ time per iteration (steps 1 through 11).

Section 5 evaluates the above algorithm for two channels, that is, for $c = 2$, and compares the *overall mean access time* achieved by the algorithm with analytical lower bounds. If a client listens to only one channel, then Equation 4 provides a lower bound ($t_{optimal}$) for the client's overall mean access time. If, however, a client listens to both channels, then the access time experienced by the client may reduce by at most a factor of 2. Therefore, a lower bound on the overall mean access time for a client listening to both channels is $t_{optimal}/2$, where $t_{optimal}$ is obtained using Equation 4.

5 Performance Evaluation

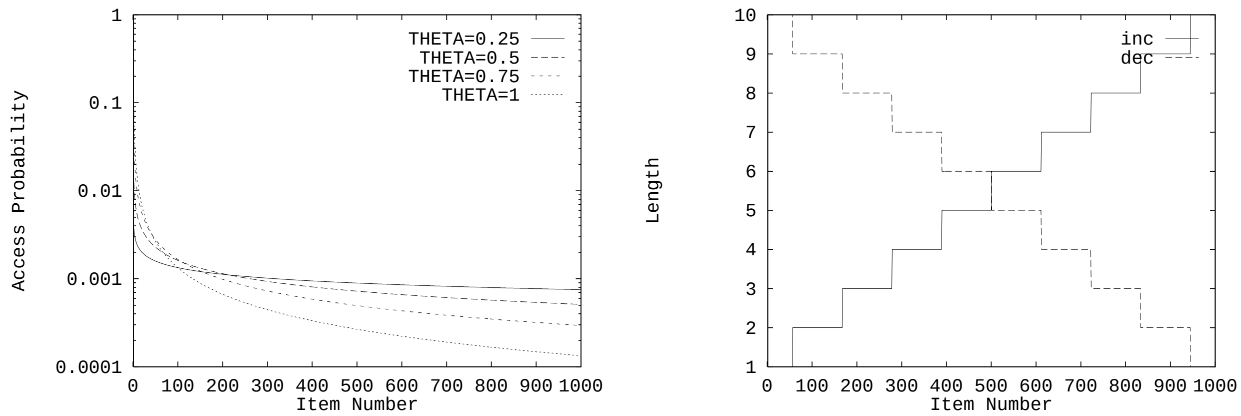
In this section, we present simulation results for various algorithms presented above. In each simulation, number of information items M is assumed to be 1000. Each simulation was conducted for at least 8 million item requests by the clients. Other parameters used in the simulation are described below.

5.1 Demand Probability Distribution

We assume that demand probabilities follow the Zipf distribution (similar assumptions are made by other researchers as well [1, 2, 3, 4, 26]). The Zipf distribution may be expressed as follows:

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

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



(a) Probability Distributions (b) Length Distributions

Figure 5: *(a) Access Probability against Item Number for various values of access skew coefficient θ used in simulation. (b) Increasing and Decreasing Length Distributions.*

5.2 Length Distribution

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

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

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

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

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

Figure 5(b) shows the two length distributions. The labels "inc" and "dec" denote Increasing and Decreasing Length distributions, respectively. In addition to these length distributions, we also use a *random* length distribution obtained by choosing lengths randomly distributed from 1 to 10 with uniform probability. Figure 6 shows the Random Length Distribution so generated. We have used these distributions in our previous work [24] as well.

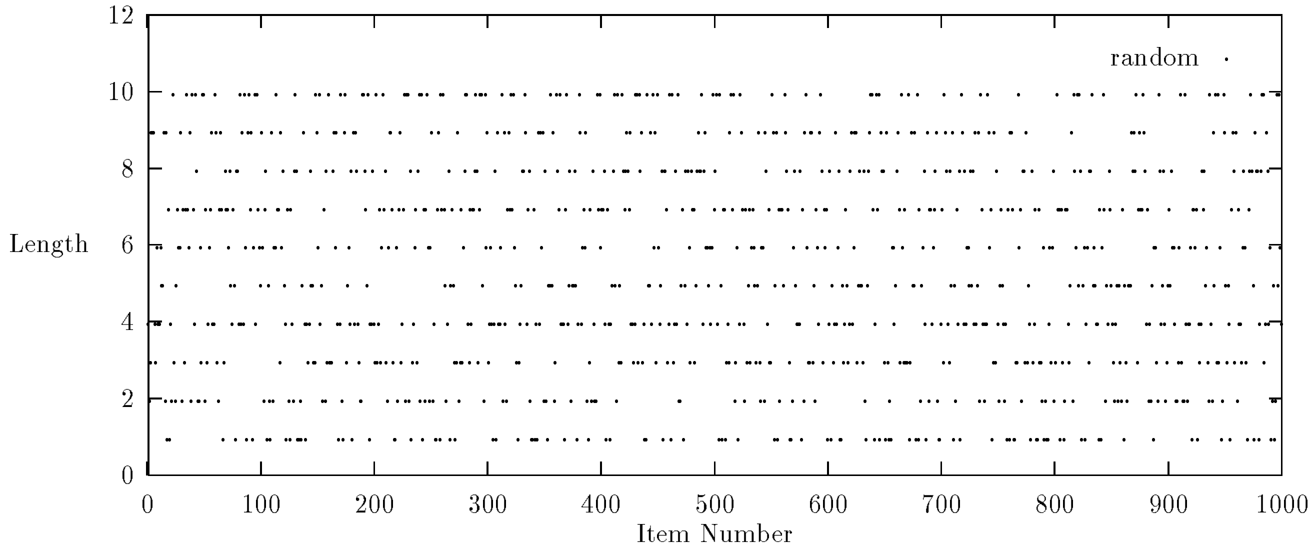


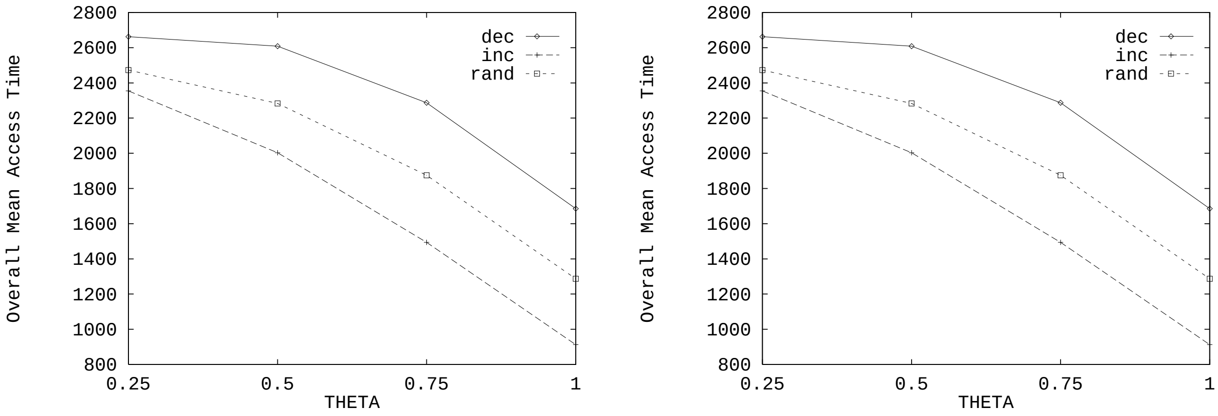
Figure 6: *Random Length Distribution generated by randomly choosing integers from 1 to 10 with uniform probability.*

5.3 Request Generation

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

5.4 Performance Evaluation for Single Channel Broadcast

In this section, we evaluate the Single Channel Scheduling Algorithm explained in Section 3. Figure 7(a) shows the simulation results. It plots *overall mean access time* against access skew coefficient θ. The curves labelled "dec", "inc" and "rand" respectively correspond to decreasing, increasing and random length distributions defined in Section 5.2. Similarly, the corresponding analytical lower bounds obtained from Equation 4 are also plot in Figure 7(b) for comparison.



(a) Simulation results (b) Analytical lower bounds

Figure 7: Overall mean access time *against access skew coefficient θ. The simulation curves are obtained using algorithm given in Section 3. The values obtained by simulation are within 0.5% of the corresponding analytical values.*

From the simulation results in Figure 7, observe that the proposed Single Channel Scheduling Algorithm performs very close to optimal. These results confirm that the algorithm is able to space instances of each item with approximately ideal spacing, thereby achieving near-optimal *overall mean access time.*

5.5 Performance Evaluation for Multi-Channel Broadcast

In this section, we evaluate performance of Multiple Channel Scheduling Algorithm explained in Section 4. We have simulated the algorithm for two channels, that is, for $c = 2$. Figures 8, 9 and 10 plot the *overall mean access time* against access skew coefficient θ for decreasing, increasing and random length distributions respectively. The curves labelled "ch1 sim" and "ch2 sim" are the curves obtained from simulation when client listens to one out of the two channels and both channels, respectively. The curves labelled "ch1 opt" and "ch2 opt" are the plot of $t_{optimal}$ and $t_{optimal}/2$ respectively where $t_{optimal}$ is obtained from Equation 4. Note that simulation results obtained for the case when the client listens to one out of two channels are different from those obtained in Section 5.4. In Section 5.4, broadcast is on single channel using algorithm in Section 3, whereas in Figures 8, 9 and 10, broadcast is on two channels using the algorithm in Section 4. Also note that for the case of two channels, the best access time that the client can observe is half of the optimal *overall mean access time* obtained for single channel case, that is, $t_{optimal}/2$. To sum up, clients listening to one channel or two channels both experience overall mean access times close to their respective lower bounds. The simulation results show that the proposed scheduling algorithm performs well for two channels.

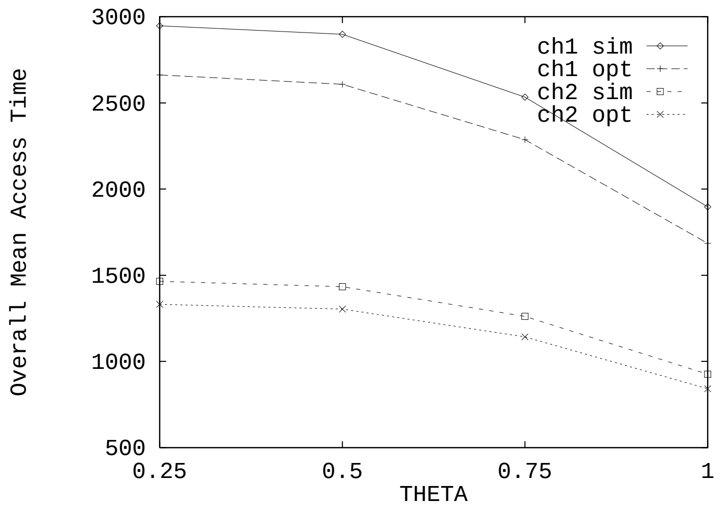


Figure 8: Overall mean access time *against access skew coefficient θ for Decreasing Length Distribution. The simulation results labelled as* sim *are within 12.6% of analytical lower bounds labelled as* opt.

6 Related Work

The algorithms presented in this paper are based on an algorithm proposed previously for "packet fair queueing" [6, 5, 18, 19]. As noted earlier, the problem of optimal broadcast scheduling is closely related to design of good packet fair queueing algorithms.

The problem of data broadcasting has received much attention lately. The existing schemes can be roughly divided into two categories (some schemes may actually belong to both categories): Schemes attempting to reduce the *access time* [4, 3, 2, 1, 13, 16, 10, 8, 26] and schemes attempting to reduce the *tuning time* [14, 15]. However, proposed on-line algorithms have not been studied previously.

Ammar and Wong [4, 26] have performed extensive research on broadcast scheduling and obtained many interesting results. One of the results obtained by Ammar and Wong is a special case of our square-root rule (Theorem 1). Wong [26] and Imielinski and Viswanathan [13, 25] present an on-line scheme that uses a *probabilistic* approach for deciding which item

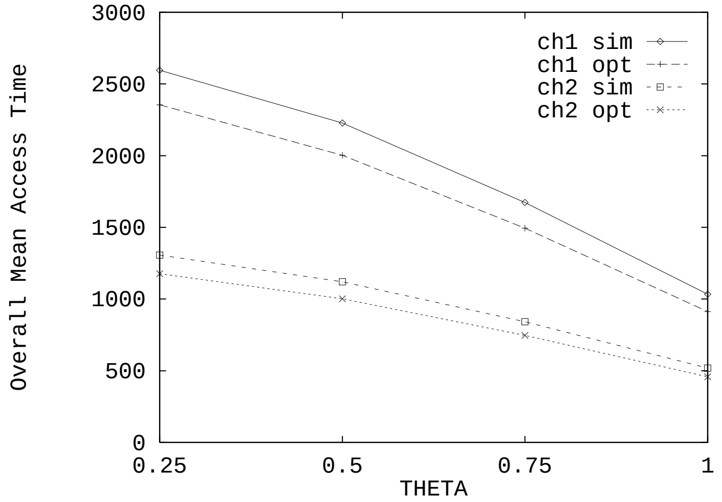


Figure 9: Overall mean access time *against access skew coefficient* θ *for Increasing Length Distribution. The simulation results labelled as* sim *are within 13.3% of analytical lower bounds labelled as* opt.

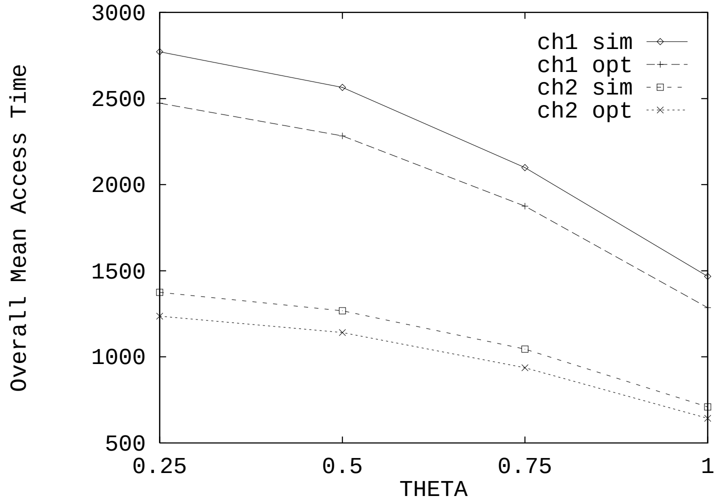


Figure 10: Overall mean access time *against access skew coefficient* θ *for Random Length Distribution. The simulation results labelled as* sim *are within 14.1% of analytical lower bounds labelled as* opt.

to transmit. The Single Channel Scheduling Algorithm presented in this report is also on-line and results in an improvement by a factor of 2 in the mean access time as compared to the probabilistic on-line algorithm in [13, 25, 26]. Chiueh [8] and Acharya et al. [3, 2, 1] present schemes that transmit the more frequently used items more often. However, they do not use optimal broadcast frequencies. Our schemes, on the other hand, tend to use optimal frequencies.

Jain and Werth [16] note that reducing the variance of spacing between consecutive instances of an item reduces the mean access time. The two schemes presented in this report do attempt to achieve a low variance. Jain and Werth [16] 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 [7]. 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).

We previously proposed algorithms [21, 24, 23] for scheduling broadcast in presence of errors, and for multiple channels. In [23], we discussed broadcasting on two channels. However, the algorithm proposed in [23] does not perform as well as Multiple Channel Scheduling Algorithm proposed in this report (Section 4) does.

Battlefield Awareness and Data Dissemination (BADD) Advanced Concept Technology Demonstration (ACTD) is a project in which our research work may be applied [9]. ACTD is managed and funded by DARPA Information System Services. The mission behind BADD project is to develop an operational system that would allow information dissemination in battlefields, maintain access to worldwide data repositories and provide tools to dynamically tailor the information system to changing battlefield situations in order to allow warfighters to view a consistent picture of the battlefield.

7 Summary

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

We propose a new on-line algorithm for scheduling broadcast on single channel called Single Channel Scheduling Algorithm, with the goal of minimizing the *access time* in asymmetric environment. The algorithm uses near-optimal frequencies for each item – these frequencies are determined as a function of item lengths, demand probability, and error rates. The proposed algorithm has $O(\log M)$ complexity which is significantly lower than a previous algorithm with comparable performance. Simulation results show that our algorithm performs quite well (very close to the theoretical optimal).

When different clients are capable of listening on different number of broadcast channels, the schedules on different broadcast channels should be designed so as to minimize the access time for all clients. The clients listening to multiple channels should experience proportionately lower delays. This report presents an algorithm for scheduling broadcasts on multiple channels called Multiple Channel Scheduling Algorithm and evaluates its performance for two channels. Simulation results show that this algorithm also performs close to optimal.

8 Future Work

The Single Channel Scheduling Algorithm, explained in Section 3, is based on the idea of Worst-case Fair Weighted Fair Queueing Algorithm (WF^2Q) presented in [5] which addresses the issue of fair allocation of bandwidth to different sessions flowing through a node on a single outgoing link. The WF^2Q algorithm tries to emulate Generalized Processor Sharing (GPS) [18] scheduling discipline. GPS is an idealized algorithm that assumes that the packets are infinitely divisible and hence cannot be implemented in practice. WF^2Q closely approximates GPS just like Single Channel Scheduling Algorithm closely approximates the ideal conditions given in Section 3. The way we used WF^2Q to schedule broadcast makes us believe that there exists some relationship between the problem of broadcast scheduling and weighted fair queueing. Currently, we are trying to map the broadcast scheduling problem to weighted fair queueing problem so that a practical solution to any one of these could easily be adapted for the other.

In addition to this, the multiple channel scheduling problem maps to the case of allocating bandwidth to different sessions flowing through a node on *multiple* outgoing links. This could be an interesting problem and to our knowledge, no one has addressed this issue yet. The mapping between the problems of multiple channel broadcast scheduling and weighted fair queueing with multiple outgoing links is one of the topics of our current research.

Acknowledgements

Thanks are due to P. Krishna for drawing our attention to the papers on packet fair queueing.

A Appendix: Proof of Theorem 1 [22]

Proof: As instances of item i are spaced equally, the spacing between consecutive instances of item i is N/f_i, where $N = \sum_{j=1}^{M} f_j l_j$ is the length of the broadcast cycle. From Equation 2, we have

$$t_{overall} = \frac{1}{2} \sum_{i=1}^{M} p_i \, s_i \tag{6}$$

Define "supply" of item i, $r_i = \frac{f_i l_i}{N}$. Thus, r_i is the fraction of time during which item i is broadcast. Now note that, $\sum_{i=1}^{M} r_i = \sum_{i=1}^{M} \frac{f_i l_i}{N} = \frac{N}{N} = 1$. Now, Equation 6 can be rewritten as,

$$t_{overall} = \frac{1}{2} \sum_{i=1}^{M} \frac{p_i l_i}{r_i} \tag{7}$$

As $\sum_{i=1}^{M} r_i = 1$, only $M - 1$ of the r_i's can be changed independently. Now, for the optimal values of r_i, we must have $\frac{\partial t_{overall}}{\partial r_i} = 0$, $\forall i$. We now solve these equations, beginning with $0 = \frac{\partial t_{overall}}{\partial r_1}$.

$$0 = \frac{\partial t_{overall}}{\partial r_1} = \frac{1}{2}\frac{\partial}{\partial r_1}\left(\sum_{i=1}^{M} \frac{p_i l_i}{r_i}\right)$$

$$= \frac{1}{2}\frac{\partial}{\partial r_1}\left(\frac{p_1 l_1}{r_1} + \sum_{i=2}^{M-1} \frac{p_i l_i}{r_i} + \frac{p_M l_M}{(1 - \sum_{i=1}^{M-1} r_i)}\right) = \frac{1}{2}\left(-\frac{p_1 l_1}{r_1^2} + \frac{p_M l_M}{(1 - \sum_{i=1}^{M-1} r_i)^2}\right)$$

$$\Longrightarrow \frac{p_1 l_1}{r_1^2} = \frac{p_M l_M}{(1 - \sum_{i=1}^{M-1} r_i)^2} \tag{8}$$

$$\text{Similarly} \quad \frac{p_2 l_2}{r_2^2} = \frac{p_M l_M}{(1 - \sum_{i=1}^{M-1} r_i)^2} \tag{9}$$

$$\tag{10}$$

From Equations 9 and 10, we get

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

$$\text{Similarly it can be shown that} \quad \frac{r_i}{r_j} = \sqrt{\frac{p_i l_i}{p_j l_j}}, \quad \forall i, j$$

This implies that, the optimal r_i must be linearly proportional to $\sqrt{p_i l_i}$. It is easy to see that constant of proportionality $a = \frac{1}{\sum_{j=1}^{M} \sqrt{p_j l_j}}$ exists such that $r_i = a\sqrt{p_i l_i}$ is the *only* possible solution for the equations $\frac{\partial t}{\partial r_i} = 0$. From physical description of the problem, we know that a non-negative minimum of t must exist. Therefore, the above solution is unique and yields the minimum t. Substituting $r_i = \frac{\sqrt{p_i l_i}}{\sum_{j=1}^{M} \sqrt{p_j l_j}}$ into Equation 7, and simplifying, yields optimal *overall mean access time* as

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

Also, the optimal frequency of item i, f_i may be obtained as $f_i = \frac{r_i N}{l_i} \propto \sqrt{p_i l_i} \frac{N}{l_i} = \sqrt{\frac{p_i}{l_i}} N$. Thus, we have shown that, optimal frequency f_i is directly proportional to $\sqrt{\frac{p_i}{l_i}}$. $\square$

B Equal-Spacing Assumption

Equation 3 provides an expression for optimal spacing between instances of an item i, $1 \leq i \leq M$. It may not be possible to achieve this spacing in reality.

Assume that number of items is $M = 3$, and cycle size $N = 6$. Let length of each item be 1. For a certain probability distribution, the optimal item frequencies and spacing are as follows: $s_1 = 2$, $s_2 = 3$, $s_3 = 6$, $f_1 = 3$, $f_2 = 2$, $f_3 = 1$.

In this case, an attempt to schedule the cycle quickly shows that, it is impossible to schedule instances of item 1 equally spaced at distance 2, and instances of item 2 equally spaced at distance 3. To do so requires that one instance of item 1 and 2 both be scheduled at the *same* time! This is called a "collision". Collisions are not permissible in a real schedule, as two items cannot be transmitted on the same channel simultaneously. This example illustrates that, in general, collisions prevent us from spacing instances of each item i equally apart.

References

[1] S. Acharya, M. Franklin, and S. Zdonik, "Prefetching from a broadcast disk," in *12th International Conference on Data Engineering*, February 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, December 1995.

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

[5] J. Bennett and H. Zhang, "Worst-case fair packet fair queueing algorithms," tech. rep., Computer Science, Carnegie Mellon University, 1996.

[6] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Sigcomm'96*, 1996.

[7] 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.

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

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

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

[11] A. Gurijala and U. Pooch, "Propagating updates in asymmetric channels (a position paper)," in *First International Workshop on Satellite-based Information Services (WOSBIS)*, November 1996.

[12] E. Horowitz and S. Sahni, *Fundamentals of Data Structures in Pascal.* Computer Science Press, Inc., 1984.

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

[14] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Power efficient filtering of data on air," in *4th International Conference on Extending Database Technology*, March 1994.

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

[16] 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.

[17] S. Keshav, "On the efficient implementation of fair queueing," *Journal of Internetworking: Research and Experience*, vol. 2, September 1991.

[18] A. K. Parekh and R. G. Gallagher, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, June 1993.

[19] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," in *SIGCOMM'95, Cambridge, MA, USA*, 1995.

[20] K. Stathatos, N. Roussopoulos, and J. S. Baras, "Adaptive data broadcasting using air-cache," in *First International Workshop on Satellite-based Information Services (WOSBIS)*, November 1996.

[21] N. H. Vaidya and S. Hameed, "Data broadcast in asymmetric environments," in *First International Workshop on Satellite-based Information Services (WOSBIS)*, November 1996.

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

[23] N. H. Vaidya and S. Hameed, "Improved algorithms for scheduling data broadcast," Tech. Rep. 96-029, Computer Science Department, Texas A&M University, College Station, December 1996.

[24] N. H. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments," Tech. Rep. 96-022, Computer Science Department, Texas A&M University, College Station, November 1996.

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

[26] J. W. Wong, "Broadcast delivery," in *Proceedings of IEEE*, pp. 1566–1577, December 1988.