Improved Algorithms for Scheduling Data Broadcast*

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

Technical Report 96-029

December 19, 1996

Abstract

With the increasing popularity of portable wireless computers, mechanisms to efficiently transmit information to such *clients* are of significant interest. The environment under consideration is *asymmetric* in that the information server has much more bandwidth available, as compared to the clients. 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. The algorithm also takes transmission errors into account. We evaluate performance of the algorithm and find it to be close to optimal. We also present an algorithm to coordinate broadcasts over two different channels, and evaluate its performance.

Key Words: Data broadcast, transmission errors, 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.

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* [22] 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, 22, 24, 25].

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, 24, 25].

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 also take transmission errors into account as wireless environments are subject to such errors. In an asymmetric environment, when a client receives an information item containing errors, it is not always possible (or desirable) for the client to request retransmission of the information. In this case, the client must wait for the next transmission of the required item. Design of optimal broadcast schedules is affected by the error rate. (Other than our work, we are not familiar with any other work that takes errors into account when scheduling broadcasts.) Previously, on-line scheduling algorithms have been proposed for error-free environments. The previous on-line algorithms either result in access times twice as large as the proposed algorithm [25, 24], or have higher time complexity of $O(M)$ [22].

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 two 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. Scheduling algorithm for broadcast on two channels is presented in Section 4. 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 Algorithm

First we introduce some terminology and notations to be used in this report.

- 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 $N = l_1 + l_2 + l_1 + l_3 = 1 + 2 + 1 + 3 = 7$.
- *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 $f_2 = f_3 = 1$.

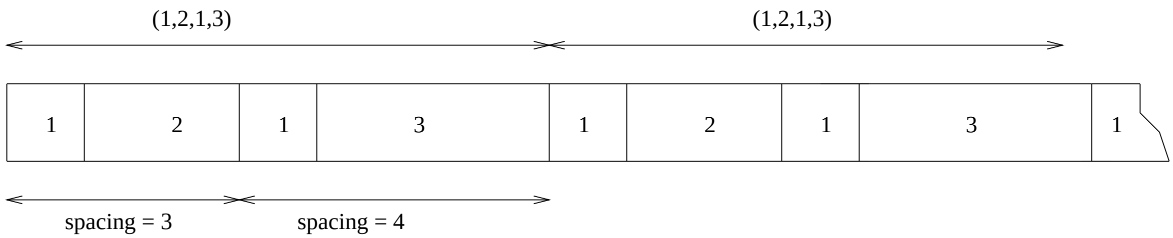


Figure 1: Broadcast schedule

- *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.

  The access time is affected by presence of errors. Forward error correction (FEC) can often correct small number of errors. However, FEC schemes cannot always correct the errors. When *uncorrectable* errors occur in an item, the contents of the item cannot be recovered. As seen below, this affects performance achieved using a given broadcast schedule.

  Now we evaluate the *item mean access time* in presence of uncorrectable errors. 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, however, the assumption does provide a basis for developing the proposed algorithm. The proposed algorithm attempts to approximate equal spacing.)

  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. If this instance does not contain uncorrectable errors, then the access time is $s_i/2$. However, if the first instance of item i transmitted after a client starts waiting contains uncorrectable errors, 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. Let the probability that an item of length l contains uncorrectable errors be denoted as $E(l)$. (In our simulations, we assume $E(l) = 1 - e^{-\lambda l}$, λ being called the error rate.) Now, the expected number of consecutive instances of item i containing uncorrectable errors is obtained as[1]

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

Then, it follows that, the *item mean access time* for item i is given by

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

Observe that the factor $\left(\frac{1+E(l_i)}{1-E(l_i)}\right)$ is an increasing function of $E(l_i)$. Thus, as intuition would suggest, environments with more noise (i.e., larger $E(l_i)$) will result in greater access times. As $E(l_i)$ is a function of l_i, it follows that the optimal broadcast frequency of item i should be a function of l_i. We later present a result that formalizes this observation.

- *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

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

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 $\sqrt{p_i \left(\frac{1+E(l_i)}{1-E(l_i)}\right)}$ and inversely proportional to $\sqrt{l_i}$. That is,*

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

[1]This expression is derived assuming that errors in different instances of the same item are independent. Wireless transmissions are subject to burst errors, making errors in consecutive *bits* correlated (not independent). However, in a typical broadcast schedule, consecutive instances of the *same* items are separated in time. Therefore, it is reasonable to assume that errors in different instances of the same item are independent.

Proof: Appendix A presents the proof. □

From Theorem 1 it follows that, there exists a constant K such that $f_i = K\sqrt{\frac{p_i}{l_i}\left(\frac{1+E(l_i)}{1-E(l_i)}\right)}$. 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

$$K = \frac{N}{\sum_{i=1}^{M} \sqrt{p_i\, l_i\left(\frac{1+E(l_i)}{1-E(l_i)}\right)}}$$

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

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

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

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

(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

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

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

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

Let ϕ_i denote the right-hand side of Equation 5. That is, $\phi_i = \frac{l_i}{\left(\sum_{j=1}^{M}\sqrt{p_j l_j\left(\frac{1+E(l_j)}{1-E(l_j)}\right)}\right)\sqrt{\frac{l_i}{p_i}\left(\frac{1-E(l_i)}{1+E(l_i)}\right)}}$. 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. It turns out that the above two conditions are similar to those imposed on "packet fair queueing" algorithms [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 [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.)

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 3: Broadcast item j at time T.
$B_j = C_j$
$C_j = B_j + s_j$
Step 4: When item j completes transmission, increment T by l_j.
Go to step 1.

The algorithm iterates steps 1 through 4 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, 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 3 (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 $p_3 = 0.25$. Also, let $E(l) = 1 - e^{-0.1\,l}$. In this case, $s_1 = 3.53$, $s_2 = 6.50$ and $s_3 = 7.32$. In the first iteration of the above algorithm, at step 2, $B_1 = B_2 = B_3 = T = 0$, and $C_1 = 3.53$, $C_2 = 6.50$ and $C_3 = 7.32$. 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, $T = 1$, $B_1 = 3.53$, $B_2 = B_3 = 0$, $C_1 = 7.06$, $C_2 = 6.50$ and $C_3 = 7.32$. Now, $S = \{2, 3\}$ (as $B_2 = B_3 = 0 < T = 1$, and $B_3 > T$). As $C_2 < C_3$, item 2 is transmitted next. Figure 2 shows the first few items transmitted using the above algorithm.

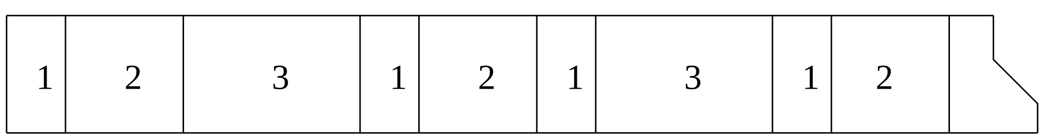


Figure 2: Illustration of the scheduling algorithm

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 [23], 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 2 channels. This algorithm is not necessarily the best algorithm for 2 channels. The algorithm is presented here as an illustration of how the algorithm in previous section may be extended for multiple channels. Design of a superior $O(c \log M)$ algorithm for c broadcast channels is a subject of on-going research.

The approach considered here uses a modification of the algorithm described in Section 3. Let the two broadcast channels be numbered 1 and 2. A client may either listen only to channel 1 (or only channel 2), or to both channels 1 and 2.

Similar to the algorithm presented above, the algorithm below maintains four variables, B_i^1, C_i^1, B_i^2 and C_i^2, for each item i.

2-Channel Broadcast Scheduling

Step 0: Determine optimal spacing s_i for each item i, using Equation 3.
Current time is denoted by T. (T is considered to be *virtual* time.) Initially, $T = 0$.
Initialize $B_i^1 = 0$ and $C_i^1 = s_i$ for $1 \leq i \leq M$.
Initialize $B_i^2 = 0$ and $C_i^2 = s_i/2$ for $1 \leq i \leq M$.

The remaining steps are executed to find an item to broadcast on channel h at time T (h may be 1 or 2).
Step 1: Determine set S of items for which $B_i^h \leq T$.
That is, $S = \{i \mid B_i^h \leq T,\ 1 \leq i \leq M\}$.
If S is empty then set T = $\min_{1 \leq i \leq M} B_i$, and go to step 1.
Step 2: Let C_{min} denote the minimum value of C_i^h 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 3: Broadcast item j at time T.
$B_j^h = B_j^h + s_j$
$C_j^h = C_j^h + s_j$
if $h = 1$ then
$B_j^2 = B_j^2 + s_j/2$
else
$B_j^1 = B_j^1 + s_j/2$
Step 4: When item j completes transmission, increment T by l_j.
Go to step 1.

This algorithm also, on average, requires $O(\log M)$ time per iteration (steps 1 through 4).

Section 5 evaluates the above algorithm, and compares the overall mean access times 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 meacn 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[2] M is assumed to be 100. Error probability $E(l)$ is assumed to be $1 - e^{-\lambda l}$ (several values of λ are considered). Each simulation for single channel 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, 25]). 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 3(a).

5.2 Length Distribution

A *length distribution* specifies length l_i of each item i. We consider two distributions in this report: (a) *Uniform Length Distribution:* In this case, $l_i = 1$, $1 \leq i \leq M$. (b) *Random Length Distribution:* In this case, integral lengths randomly distributed from 1 to 10 are assigned to the items with uniform probability. The *random length distribution* is shown in Figure 3(b).

[2]Our past experiments suggest that broadcast scheduling algorithms tend to be more sub-optimal for smaller number of items. Therefore, we choose M relatively small (to attempt to show the worst-case behavior of the algorithms). Besides, smaller M also allowed us to complete the simulations in a shorter duration of time.

5.3 Request Generation

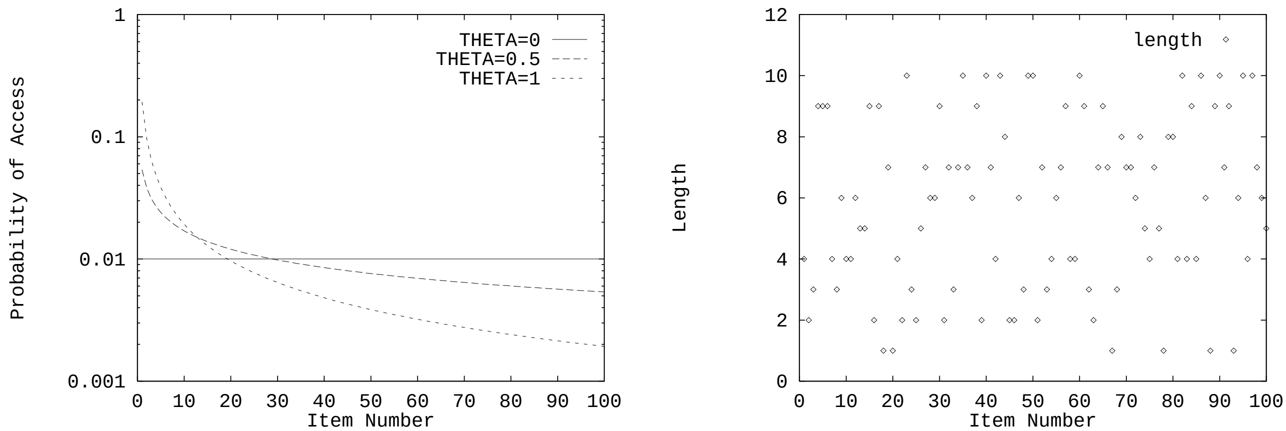
For our simulations, we generated 2 requests for items per time unit. Simulation time is divided into intervals of unit length; 2 requests are generated during each such interval. The time at which the requests are made is uniformly distributed over the corresponding unit length interval. The items for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

5.4 Performance Evaluation for Single Channel Broadcast

In this section, we evaluate performance of the on-line algorithm for single channel as explained in Section 3. Figures 4 and 5 plot *overall mean access time* in the presence of errors for different error rates (λ), and for random and uniform length distributions, respectively. In each of these figures, part (a) plots the simulation results and part (b) plots analytical lower bounds for $\theta = 0, 0.5$ and 1 (the lower bounds are obtained using Equation 4, substituting $E(l) = 1 - e^{-\lambda l}$). From the simulation results, observe that the proposed on-line algorithm achieves performance very close to optimal. As expected, the access time increases with increasing error rate λ. However, observe that the proposed algorithm performs close to optimal for all values of λ. The simulation results confirm that the proposed algorithm is able to space instances of each item with approximately ideal spacing, thereby achieving near-optimal *overall mean access time*. Other researchers do not take uncorrectable errors into account when determining the broadcast schedules.

5.5 Performance Evaluation for 2-Channel Broadcast

In this section, we evaluate performance of the on-line algorithm for two channels as explained in Section 4. Figure 6 plots *overall mean access time* for a client which listens to only one channel, for different error rates (λ) and random length distribution. Figure 7 plots *overall mean access time* for a client listening to both the channels. Again, in each of these figures, part (a) plots the simulation results and part (b) plots analytical lower bounds for $\theta = 0, 0.5$ and 1. The lower bound for a client listening to a single channel is $t_{optimal}$ (obtained using Equation 4) and the lower bound for a client listening to both channels is $t_{optimal}/2$. From the simulation results, observe that the proposed algorithm achieves performance quite close to optimal. Clients listening to one channel or two channels both experience overall mean access times close to the respective lower bounds. The simulation results show that the proposed scheduling algorithm for 2 channels performs well. Extension of this algorithm to arbitrary number of channels is a subject of current research.

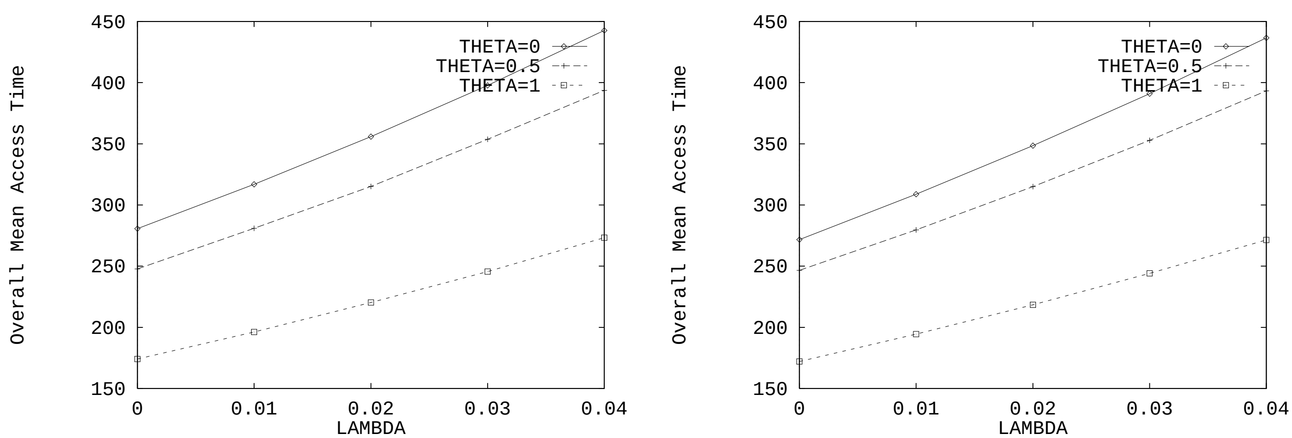


(a) Zipf probability distribution

(b) Random length distribution

Figure 3: (a) Zipf distribution for various values of access skew coefficient θ. Note that the scale on vertical axis is logarithmic. (b) *Random* length distribution. Lengths are chosen as integers between 1 and 10 with uniform probability.

With Random Length Distribution

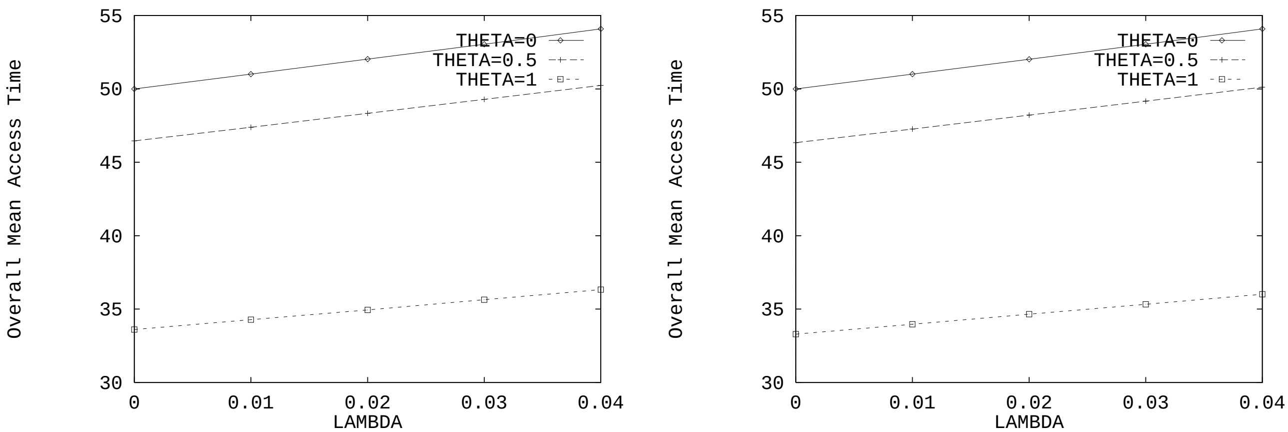


(a) Simulation results

(b) Analytical lower bounds

Figure 4: *Overall mean access time* against λ for different values of θ and random length distribution. The simulation curves are obtained using algorithm given in Section 3.

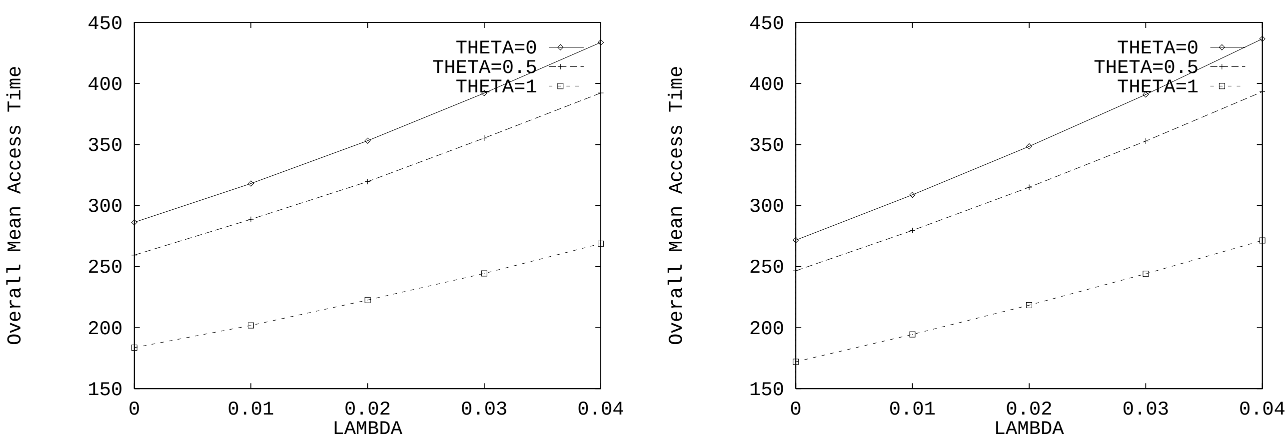
With Uniform Length Distribution



(a) Simulation results (b) Analytical lower bounds

Figure 5: *Overall mean access time* against λ for different values of θ and uniform length distribution. The simulation curves are obtained using algorithm given in Section 3.

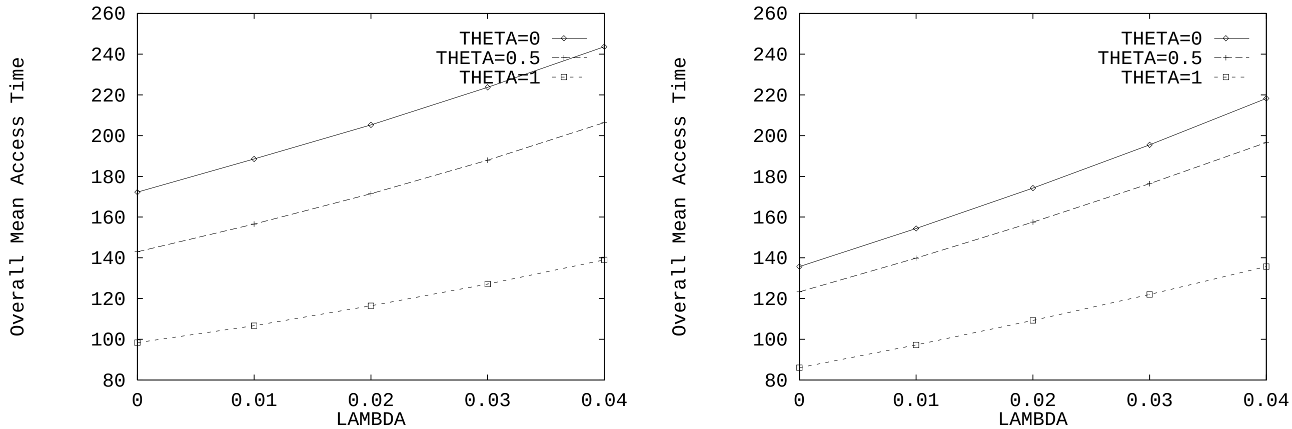
With Random Length Distribution and Client listening to 1 channel



(a) Simulation results (b) Analytical lower bounds

Figure 6: *Overall mean access time* against λ for different values of θ and random length distribution observed by a client listening to only one of the two broadcast channels. The simulation curves are obtained using algorithm given in Section 4.

With Random Length Distribution and Client listening to 2 channels



(a) Simulation results

(b) Analytical lower bounds

Figure 7: *Overall mean access time* against λ for different values of θ and random length distribution observed by a client listening to both the channels. The simulation curves are obtained using algorithm given in Section 4.

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, 25] and schemes attempting to reduce the *tuning time* [14, 15]. However, proposed on-line algorithms have not been studied previously. Also, impact of errors on scheduling and multiple-channel broadcast have not been addressed by other researchers.

For error-free environments, Ammar and Wong [4, 25] 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 [25] and Imielinski and Viswanathan [13, 24] present an on-line scheme that uses a *probabilistic* approach for deciding which item to transmit. Our on-line algorithm results in an improvement by a factor of 2 in the mean access time as compared to the probabilistic on-line algorithm in [13, 24, 25]. 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 [22] for scheduling broadcast in presence of errors, and for multiple channels. However, our previous algorithms had a higher time complexity as compared to the algorithms presented in this report.

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 broadcasts, with the goal of minimizing the *access time* in an environment that may be subject to errors. 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 algorithms perform 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 2 channels. Simulation results show that this algorithm also performs close to optimal.

Acknowledgements

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

A Appendix: Proof of Theorem 1

Proof: As instances of item i are spaced equally, the spacing between consecutive instances of item i is N/f_i, where $N = \sum_{j=1}^{M} f_j l_j$ is the length of the broadcast cycle. Let, Z_i denote $p_i \left(\frac{1+E(l_i)}{1-E(l_i)}\right)$. Then, from Equation 2, we have

$$t_{overall} \;=\; \frac{1}{2}\sum_{i=1}^{M} p_i\, s_i \left(\frac{1+E(l_i)}{1-E(l_i)}\right) \;=\; \frac{1}{2}\sum_{i=1}^{M} Z_i\, s_i \tag{6}$$

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,

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

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 $0 = \frac{\partial t_{overall}}{\partial r_1}$.

$$\begin{aligned}
0 \;&=\; \frac{\partial t_{overall}}{\partial r_1} \;=\; \frac{1}{2}\frac{\partial}{\partial r_1}\left(\sum_{i=1}^{M}\frac{Z_i l_i}{r_i}\right) \\
&=\; \frac{1}{2}\frac{\partial}{\partial r_1}\left(\frac{Z_1 l_1}{r_1} + \sum_{i=2}^{M-1}\frac{Z_i l_i}{r_i} + \frac{Z_M l_M}{(1-\sum_{i=1}^{M-1} r_i)}\right) \;=\; \frac{1}{2}\left(-\frac{Z_1 l_1}{r_1^2} + \frac{Z_M l_M}{(1-\sum_{i=1}^{M-1} r_i)^2}\right)
\end{aligned}$$

$$\Longrightarrow \frac{Z_1 l_1}{r_1^2} \;=\; \frac{Z_M l_M}{(1-\sum_{i=1}^{M-1} r_i)^2} \tag{8}$$

$$\text{Similarly} \quad \frac{Z_2 l_2}{r_2^2} \;=\; \frac{Z_M l_M}{(1-\sum_{i=1}^{M-1} r_i)^2} \tag{9}$$

$$\tag{10}$$

From Equations 9 and 10, we get

$$\frac{Z_1 l_1}{r_1^2} = \frac{Z_2 l_2}{r_2^2} \;\Longrightarrow\; \frac{r_1}{r_2} \;=\; \sqrt{\frac{Z_1 l_1}{Z_2 l_2}}$$

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

This implies that, the optimal r_i must be linearly proportional to $\sqrt{Z_i l_i}$. It is easy to see that constant of proportionality $a = \frac{1}{\sum_{j=1}^{M}\sqrt{Z_j l_j}}$ exists such that $r_i = a\sqrt{Z_i l_i}$ 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 $r_i = \frac{\sqrt{Z_i l_i}}{\sum_{j=1}^{M} \sqrt{Z_j l_j}}$ into Equation 7, and simplifying, yields optimal *overall mean access time* as

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

Also, the optimal frequency of item i, f_i may be obtained as $f_i = \frac{r_i N}{l_i} \propto \sqrt{Z_i l_i} \frac{N}{l_i} = \sqrt{\frac{Z_i}{l_i}} N$. Thus, we have shown that, optimal frequency f_i is directly proportional to $\sqrt{\frac{Z_i}{l_i}}$. Theorem 1 is obtained by substituting $Z_i = p_i \left(\frac{1+E(l_i)}{1-E(l_i)} \right)$ in the above proportionality. $\square$

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 and error rate, 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 wireless environments," in *Workshop on Satellite Based Information Services (WOSBIS), Rye, NY*, November 1996.

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

[23] 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.

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

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