Efficient algorithms for scheduling data broadcast

Sohail Hameed and Nitin H. Vaidya

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

With the increasing acceptance of wireless technology, mechanisms to efficiently transmit information to wireless clients are of 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. This paper makes the simple, yet useful, observation that the problem of broadcast scheduling is related to the problem of *fair queueing*. Based on this observation, we present a log-time algorithm for scheduling broadcast, derived from an existing fair queueing algorithm. This algorithm significantly improves the time-complexity over previously proposed broadcast scheduling algorithms. Modification of this algorithm for transmissions that are subject to errors is considered. Also, for environments where different users may be listening to different number of broadcast channels, we present an algorithm to coordinate broadcasts over different channels. Simulation results are presented for proposed algorithms.

1. Introduction

With the increasing acceptance of wireless technology, mechanisms to efficiently transmit information to wireless clients are of 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 [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, several researchers have proposed algorithms for designing *broadcast schedules* [1–3,5–8,14–16,18–20,22,24–27,36,37,39].

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 [1,2,5,15,18,19,24–27,37,39]. This paper makes three contributions:

- We observe that the problem of broadcast scheduling is related to *packet fair queueing* [10,28,30]. While obvious in the hindsight, this observation has not been exploited before to design efficient broadcasting algorithms.
- Based on the above observation, we present a O(log M) broadcast scheduling algorithm, where M is the number of information items. Simulations show that this algorithm achieves near-optimal performance. A modification of the algorithm to take transmission errors into account is also presented.
- 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 above algorithm to such an environment.

The rest of this paper is organized as follows. Section 2 introduces terminology, and derives some theoretical results that motivate the proposed algorithms. Section 3 compares packet fair queueing and broadcast scheduling. Section 4 presents the proposed scheduling algorithm for a single channel. Section 5 shows how the proposed algorithm can be modified to take transmission errors into account. Section 6 presents scheduling algorithms for broadcast on two and three channels. Section 7 evaluates the performance of our algorithms. Related work is discussed in section 8. A summary is presented in section 9.

2. Preliminaries

Database at the server is assumed to be divided into many *information items*. l_i represents the 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 denotes the total number of information items in the server's database. The items are numbered 1 through M. An appearance of an item in the broadcast is referred to as an *instance* of the item.

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. It can be shown that, for optimal broadcast scheduling, all instances of an item should be equally spaced [27,33]. Hereafter, for our theoretical development, we assume that all instances of item i are spaced s_i apart. The equal-spacing assumption cannot always be realized in practice [34], however, the assumption does provide a basis for developing the proposed algorithms.

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. Arrival of client requests is assumed to be governed by a Poisson process. From the Poisson process assumption, it follows that [4] 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, $t_i = s_i/2$.

Demand probability of item i, denoted as p_i, is the probability that an item needed by a client is item i. *Overall Mean Access Time*, denoted as t_{overall}, is defined as the average wait encountered by a client (averaged over all items). Thus, $t_{\text{overall}} = \sum_{i=1}^{M} p_i t_i$. Substituting $t_i = s_i/2$, we get

$$t_{\text{overall}} = \frac{1}{2}\sum_{i=1}^{M} p_i s_i. \qquad (1)$$

The theorem below provides a theoretical basis for the proposed scheduling schemes. The proof is presented in [33,35].

Theorem 1. Assuming that instances of each item i are equally spaced with spacing s_i, minimum *overall mean access time* is achieved when s_i is given by

$$s_i = \left(\sum_{j=1}^{M} \sqrt{p_j l_j}\right)\sqrt{\frac{l_i}{p_i}}. \qquad (2)$$

Substituting this expression for s_i into equation (1), the optimal *overall mean access time*, named t_{optimal}, is obtained as

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

t_{optimal} is derived assuming that instances of each item are equally spaced. As noted before, the *equal-spacing* assumption cannot always be realized. Therefore, t_{optimal} represents a *lower bound* on the overall mean access time. The lower bound, in general, is not achievable. However, as shown later, it is often possible to achieve *overall mean access time* almost identical to the above lower bound.

3. Broadcast scheduling and packet fair queueing

Consider a switch that has many input channels (queues), but just one output channel, as shown in figure 1. *Packet fair queueing* algorithms [10,30] determine which packet from the many input queues should be transmitted next on the output channel. Packet fair queueing algorithms typically attempt to satisfy two conditions:

- For a specified value ϕ_i, input queue i should get at least fraction ϕ_i of the output bandwidth (if the input queue is non-empty), such that $\sum_i \phi_i = 1$. When all input queues are non-empty, this requirement reduces to allocating exactly ϕ_i fraction of the bandwidth to input queue i. For instance, in figures 1(a) and (b) input queue 1 gets half the output bandwidth, and the other two queues get $1/4$ of the bandwidth.
- Bandwidth allocated to a particular input queue should be "evenly distributed", rather than *bursty*. For instance, while in figures 1(a) and (b) the first input queue receives half the bandwidth, the situation in figure 1(a) is preferred over (b), because in (a), packets from input queue 1 are distributed evenly on the output channel.

Now consider broadcast scheduling. As noted previously, for an optimal schedule, the spacing between consecutive instances of item i should be obtained using equation (2). From equation (2), we get

$$\frac{l_i}{s_i} = \frac{l_i}{\left(\sum_{j=1}^{M}\sqrt{p_j l_j}\right)\sqrt{l_i/p_i}} = \frac{\sqrt{p_i l_i}}{\sum_{j=1}^{M}\sqrt{p_j l_j}}. \qquad (4)$$

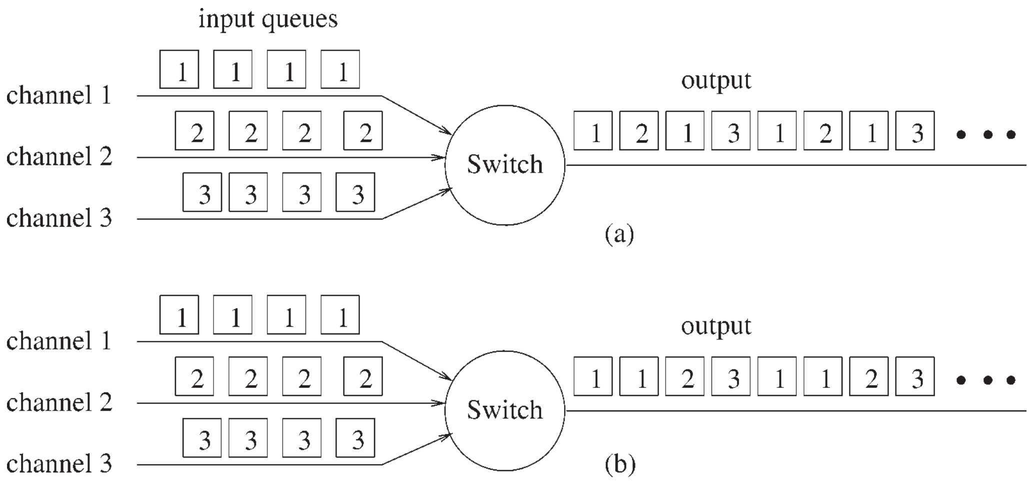


Figure 1. Packet fair queueing.

Let ϕ_i denote the right-hand side of equation (4). That is, $\phi_i = \sqrt{p_i l_i}/\sum_{j=1}^{M}\sqrt{p_j l_j}$. Then, we have $l_i/s_i = \phi_i$, and $\sum_{i=1}^{M}\phi_i = 1$. Thus, the two conditions for obtaining an optimal broadcast schedule are:

- $l_i/s_i = \phi_i$ for each item i. Observe that l_i/s_i is the fraction of broadcast channel bandwidth allocated to item i. This is similar to the requirement for fair queueing that input channel i should receive fraction ϕ_i of the output bandwidth.
- All instances of each item i should be spaced equally apart with spacing s_i. This is similar to the "even distribution" requirement of fair queueing [10].

Although the problem of *packet fair queueing* is not identical to broadcast scheduling, above similarities motivated us to adapt a packet fair queueing algorithm in [10] to broadcast scheduling. The broadcast scheduling algorithm, thus obtained, is presented below.

4. Single channel broadcast scheduling scheme

In this section, we consider the case when the information items are broadcast on a single channel. Section 6 considers multiple channel broadcast. 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 (2).
        Current time is denoted by $T$. Initially, $T = 0$.
        Initialize $B_i = 0$ and $C_i = s_i$ for $1 \leqslant i \leqslant M$.
Step 1. Determine set $S$ of items for which $B_i \leqslant T$.
        That is, $S = \{i \mid B_i \leqslant T,\ 1 \leqslant i \leqslant M\}$.
        (It can be shown that $S$ is never empty.)
Step 2. Let $C_{\min}$ = minimum value of $C_i$ over $i \in S$.
Step 3. Choose any one item $j \in S$ such that $C_j = C_{\min}$.
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, $T = T + l_j$.
        Go to step 1.
```

The algorithm iterates steps 1–5 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. As shown in appendix A, steps 1–4 can be implemented such that the average time complexity per iteration is O($\log M$).

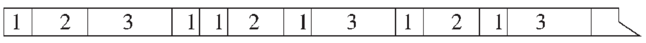


Figure 2. Illustration of the single channel scheduling algorithm.

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, by equation (2), $s_1 = 3.224$, $s_2 = 6.448$ 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, $S = \{1, 2, 3\}$, because $T = 0$, 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. Note that, after an initial transient phase, the schedule becomes cyclic with the cycle being (1,2,1,3).

5. Impact of transmission errors on scheduling

In the discussion so far, we assumed that each item transmitted by the server is always received correctly by each client. When the wireless medium is subject to transmission errors, this assumption is not always valid. Traditionally, in an environment that is subject to errors, 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 received item must be discarded [33]. Thus, 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.

Suppose that uncorrectable errors occur in an item of length l with probability $E(l)$. Now, l_i denotes the length of item i after encoding with an error control code. It can be shown that theorem 1 needs to be modified to take errors into account as follows. We omit the proof here due to lack of space [21,35].

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

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

Observe that when errors do not occur, $E(l) = 0$, and equation (5) reduces to equation (2). With transmission errors, the optimal *overall mean access time*, named $t_{\text{opt\_err}}$, is as follows [33]:

$$t_{\text{opt\_error}} = \frac{1}{2}\left(\sum_{i=1}^{M}\sqrt{p_i l_i \frac{1+E(l_i)}{1-E(l_i)}}\right)^2. \qquad (6)$$

5.1. Scheduling algorithm with transmission errors

The on-line scheduling algorithm presented in the previous section can be used with transmission errors, with the only modification that, in step 0 of the algorithm, the value of s_i is obtained using equation (5), instead of equation (2). No other changes are made in the algorithm. The modified algorithm is evaluated in section 7.

6. 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).

To illustrate how the algorithm in section 4 may be extended for multiple channels, we present algorithms for scheduling broadcast on two and three channels. In practice, we do not expect a client to be capable of listening to too many channels simultaneously.

Assume that the broadcast channels are numbered from 1 to c, where c is the number of channels. We assume that a client listening to j channels, $1 \leqslant j \leqslant c$, must listen to first j consecutive channels. Thus, a client listening to, say, 2 channels must listen to channels 1 and 2. Let π_j denote the probability that a client listens to j channels. Trivially, $\sum_{j=1}^{c}\pi_j = 1$.

6.1. Optimality criteria

For single channel scheduling, we attempted to minimize overall mean access time, t_{overall}. However, with multiple channels, the overall mean access time experienced by clients listening to different number of channels would be different. Let $t_{\text{overall}(i)}$ denote the overall mean access time experienced by clients listening to the first i channels. Then, the performance metric of interest here, called *composite* overall mean access time, denoted as $t_{\text{composite\_overall}}$, is obtained as

$$t_{\text{composite\_overall}} = \sum_{i=1}^{c}\pi_i t_{\text{overall}(i)}. \qquad (7)$$

When a client listens to only 1 channel, a lower bound on the overall mean access time $t_{\text{overall}(1)}$ is given by t_{optimal} in equation (3). It is easy to see that a lower bound on $t_{\text{overall}(i)}$ is given by t_{optimal}/i. Thus, a lower bound on $t_{\text{composite\_overall}}$ can be obtained as

$$t_{\text{composite\_optimal}} = \sum_{i=1}^{c}\pi_i \frac{t_{\text{optimal}}}{i}. \qquad (8)$$

The objective now is to design multi-channel algorithms that minimize $t_{\text{composite\_overall}}$.

6.2. Staggered broadcast schedules

The main idea here is to schedule broadcast of an item i in such a way that its instances on consecutive channels are “staggered” with some interval. As an example, figure 3 shows scheduling of an item i on three channels. The instances on channel 2 are staggered by an interval of ψ_{i2}, and those on channel 3 are staggered by an interval of ψ_{i3}, with respect to the corresponding instances on channel 1. Note that the spacing between instances of item i on each channel is s_i.

If we assume that every client is listening to all the three channels, i.e., $\pi_3 = 1$, $\pi_1 = \pi_2 = 0$, then clearly, $\psi_{i3} = 2\psi_{i2} = \frac{2}{3}s_i$ would be optimal. With these values, instances of item i are staggered across the three channels such that a client listening to three channels would receive item i every $s_i/3$ time units. In general, however, optimal ψ_{i2} and ψ_{i3} would vary with different π_j distributions.

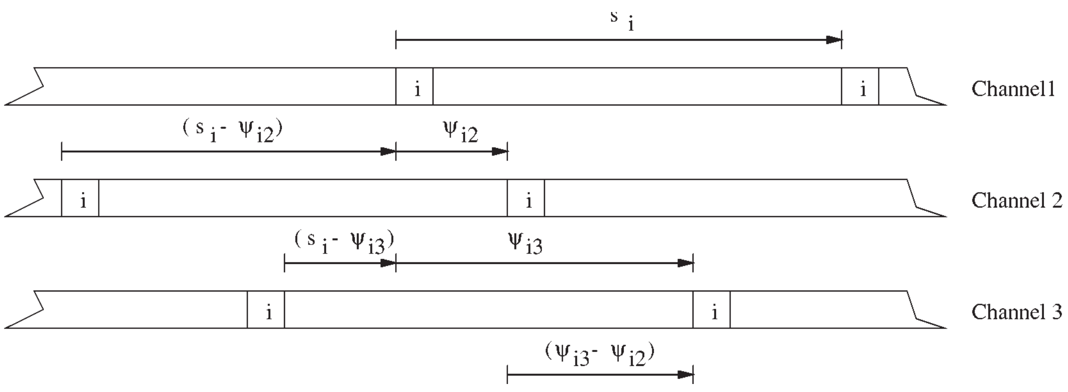


Figure 3. Schedule for item i on three channels. The instances of item i on channel 2 are staggered by an interval of ψ_{i2} and on channel 3 by an interval of ψ_{i3} with respect to channel 1.

6.3. 2-channel scheduling

Let us consider the case when $c = 2$. Hence a client either listens only to channel 1, or to both channels. Appendix B.1 shows that, for optimality, $\psi_{i2} = s_i/2$. Note that the value of ψ_{i2} is independent of the values of π_1 and π_2. That is, every instance of item i on channel 2 should appear exactly midway between every two consecutive instances of item i on channel 1, independent of the values of π_1 and π_2. (As seen later, for 3 channels, amount of stagger depends on π distribution.)

Similar to single channel scheduling, the proof of the above result assumes that the consecutive instances of all items are equally spaced on a given channel. In addition, the proof also assumes that an instance of item i on channel 2 appears exactly after ψ_{i2} time units from an instance on channel 1. These assumptions may not be realizable in general. However, they provide a theoretical foundation on which an algorithm may be developed.

The following algorithm tries to achieve optimal staggering for 2 channels. Similar to the algorithm presented in the previous section, for item i, the algorithm below maintains B_i^j and C_i^j, for channel j, $j = 1, 2$.

2-channel broadcast scheduling

```
Step 0. Determine optimal spacing $s_i$ for each item $i$, using
        equation (2) (or equation (5), if transmission errors
        can occur).
        Current time is denoted by $T$. Initially, $T = 0$.
        Initialize $B_i^1 = B_i^2 = 0$ and $C_i^1 = C_i^2 = s_i$, $1 \leqslant i \leqslant M$.
```

Steps below are executed to find an item to transmit on channel h at time T (h may be 1 or 2).

```
Step 1. Determine set $S$ of items for which $B_i^h \leqslant T$.
        That is, $S = \{i \mid B_i^h \leqslant T,\ 1 \leqslant i \leqslant M\}$.
Step 2. Let $C_{\min}$ = minimum value of $C_i^h$ over $i \in S$.
Step 3. Choose any one item $j \in S$ such that $C_j^h = C_{\min}$.
Step 4. Broadcast item $j$ at time $T$.
        If $h = 1$ then {
            $C_j^2 = T + \psi_{j2}$,
            $B_j^2 = C_j^2 - s_j$}.
        $B_j^h = B_j^h + s_j$,
        $C_j^h = B_j^h + s_j$.
```

Steps 1–4, on average, require O($\log M$) time, similar to the algorithm in section 4.

6.4. 3-channel scheduling

Unlike in case of $c = 2$, for three channels ($c = 3$), optimal values of ψ's are dependent on π's. Appendix B.2 shows that, for optimality with 3 channels,

$$\psi_{i2} = \frac{2\pi_2 + \pi_3}{4\pi_2 + 3\pi_3} s_i, \tag{9}$$

$$\psi_{i3} = \frac{3\pi_2 + 2\pi_3}{4\pi_2 + 3\pi_3} s_i. \tag{10}$$

The 2-channel algorithm above can be modified for 3 channels, as follows:

3-channel broadcast scheduling

```
Step 0. Determine optimal spacing $s_i$ for each item $i$, using
        equation (2) (or equation (5), if transmission errors
        can occur).
        Current time is denoted by $T$. Initially, $T = 0$.
        Initialize $B_i^1 = B_i^2 = B_i^3 = 0$ and
            $C_i^1 = C_i^2 = C_i^3 = s_i$ for $1 \leqslant i \leqslant M$.
        Determine $\psi_{ij}$, $j = 2, 3$ and $1 \leqslant i \leqslant M$.
```

Steps below are executed to find an item to broadcast on channel h at time T (h may be 1, 2 or 3).

```
Step 1. Determine set $S$ of items for which $B_i^h \leqslant T$.
        That is, $S = \{i \mid B_i^h \leqslant T,\ 1 \leqslant i \leqslant M\}$.
Step 2. Let $C_{\min}$ = minimum value of $C_i^h$ over $i \in S$.
Step 3. Choose any one item $j \in S$ such that $C_j^h = C_{\min}$.
Step 4. Broadcast item $j$ at time $T$.
        If $h = 1$ then {
            $C_j^2 = T + \psi_{j2}$,
            $B_j^2 = C_j^2 - s_j$,
            $C_j^3 = T + \psi_{j3}$,
            $B_j^3 = C_j^3 - s_j$}
        else if $h = 2$ then {
            $C_j^3 = T + (\psi_{j3} - \psi_{j2})$,
            $B_j^3 = C_j^3 - s_j$}.
        $B_j^h = B_j^h + s_j$,
        $C_j^h = B_j^h + s_j$.
```

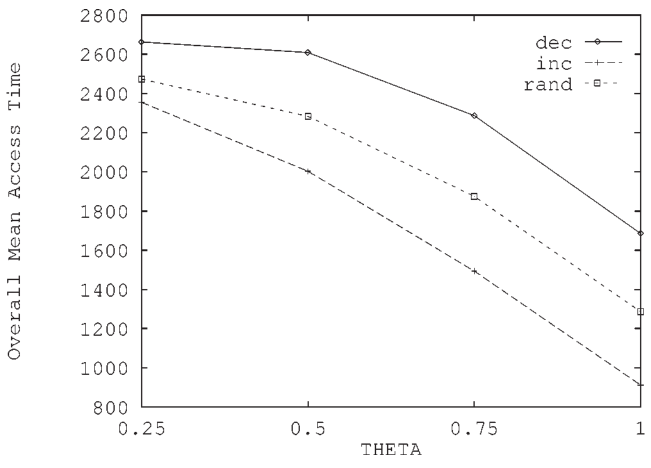
Steps 1–4, on average, require O($\log M$) time, similar to the algorithm in section 4. This algorithm can be extended for $c > 3$.

7. Performance evaluation

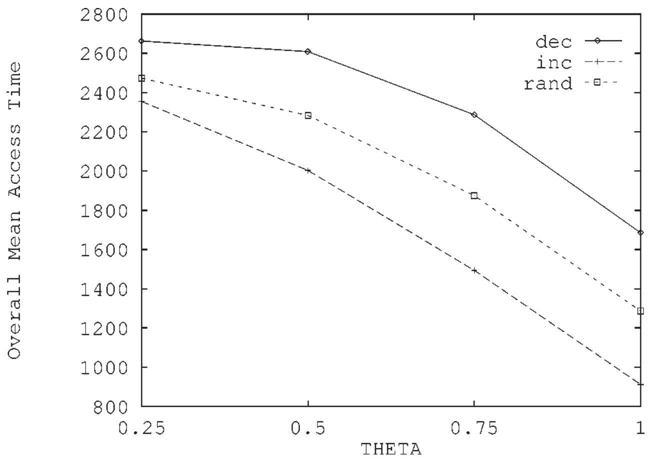
In this section, we present simulation results for various algorithms presented above. In each simulation, the number of information items M is assumed to be 1000. Each simulation was conducted for at least 8 million item requests by the clients. We assume that demand probabilities follow the Zipf distribution (similar assumptions are made by other researchers as well [2,5,37]). The Zipf distribution may be expressed as

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

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

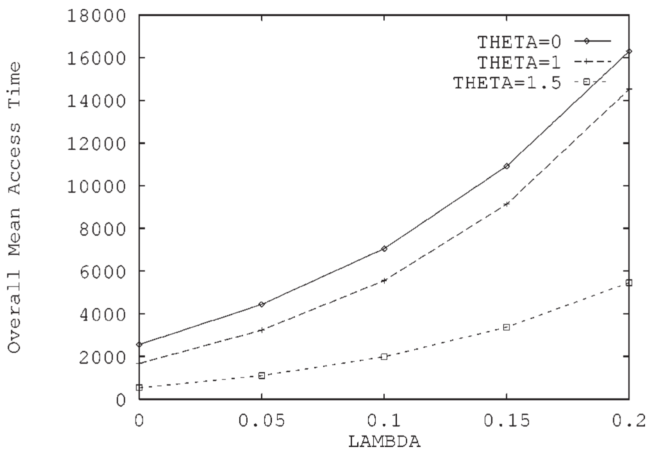


(a) Simulation results

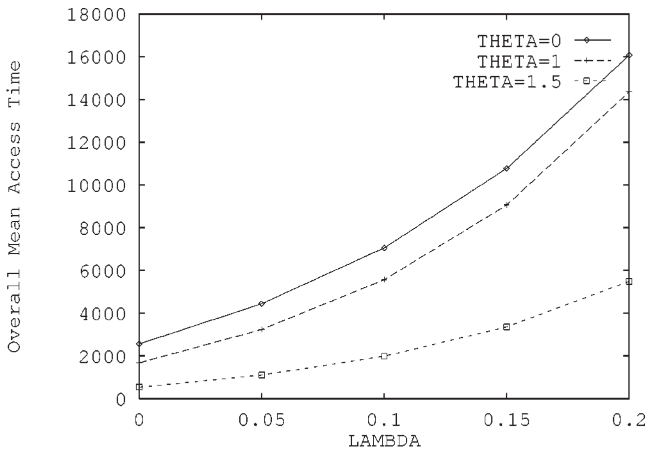


(b) Analytical lower bounds

Figure 4. Overall mean access time versus access skew coefficient θ. The simulation curves are obtained using the algorithm given in section 4. The values obtained by simulation are within 0.5% of the corresponding analytical values.



(a) Simulation results



(b) Analytical lower bounds

Figure 5. Overall mean access time versus error rate λ for different values of θ and decreasing length distribution. The simulation curves are obtained using the single channel scheduling algorithm modified to take errors into account. The simulation results are within 1.1% of analytical lower bounds.

distribution becomes increasingly “skewed” as θ increases (that is, for larger θ, the range of p_i values becomes larger).

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

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

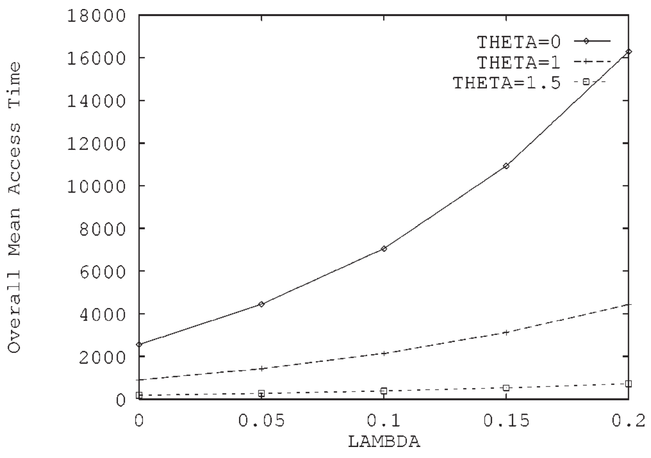
where L_0 and L_1 are parameters that characterize the distribution. L_0 and L_1 are both non-zero integers. The `round()` function above returns a rounded integer value of its argument. We consider two special cases of the above length distribution: (i) *Increasing Length Distribution* obtained by $L_0 = 1$ and $L_1 = 10$, and (ii) *Decreasing Length Distribution* obtained by $L_0 = 10$ and $L_1 = 1$. In addition to above length distributions, we also use a *Random Length Distribution* obtained by choosing lengths randomly distributed from 1 to 10 with uniform probability.

We generated two requests per time unit. The items for which requests are made are determined using the demand probability distribution.

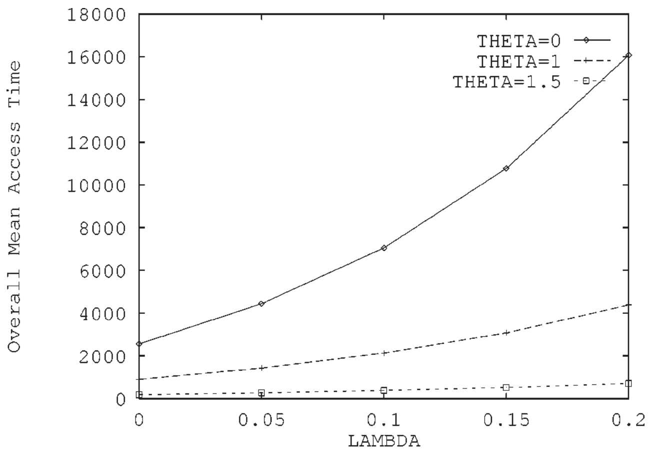
7.1. Performance evaluation for single channel broadcast

7.1.1. Single channel broadcast without transmission errors

In this section, we evaluate the single channel scheduling algorithm in section 4 (assuming that transmission errors do not occur). Figure 4(a) shows the simulation results. It plots *overall mean access time* versus access skew coefficient θ. The curves labeled “dec”, “inc” and “rand”, respectively, correspond to decreasing, increasing and random length distributions defined above. The corresponding analytical lower bounds obtained from equation (3) are plotted in figure 4(b) for comparison. From the simulation results in figure 4, observe that the proposed single channel scheduling algorithm performs very close to optimal (within 0.5% of 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*.

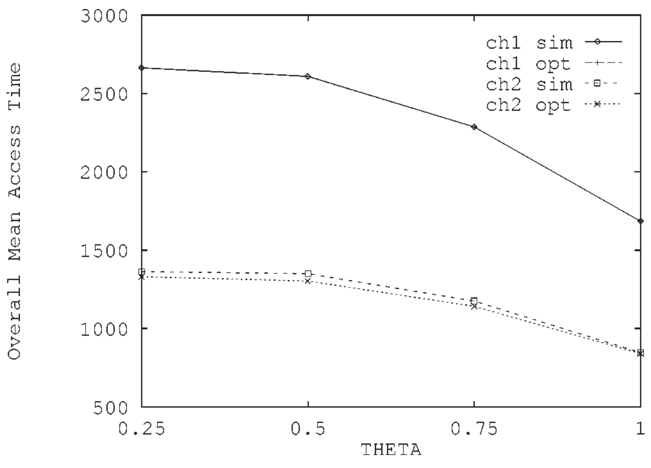


(a) Simulation results

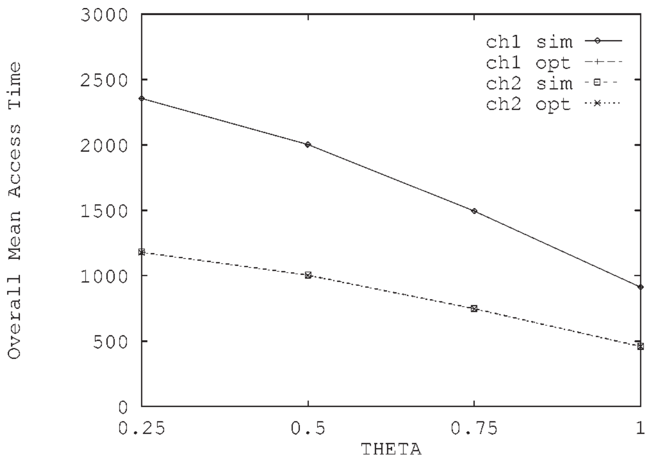


(b) Analytical lower bounds

Figure 6. Overall mean access time versus λ for different values of θ and increasing length distribution. The simulation curves are obtained using the single channel scheduling algorithm modified to take errors into account. The simulation results are within 2.5% of analytical lower bounds.



(a) For Decreasing Length



(b) For Increasing Length

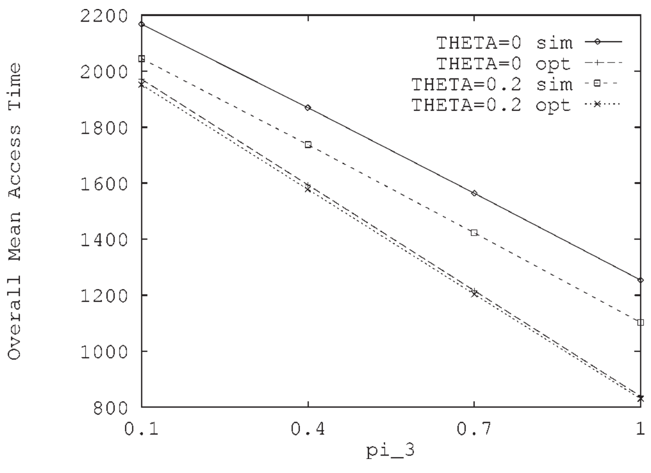
Figure 7. Overall mean access time versus access skew coefficient θ for (a) decreasing length and (b) increasing length distributions. The simulation results labeled as `sim` are within 3.6% of analytical lower bounds labeled as `opt`. Note that the curves `ch1 sim` and `ch1 opt` are overlapping.

7.1.2. Single channel broadcast with transmission errors

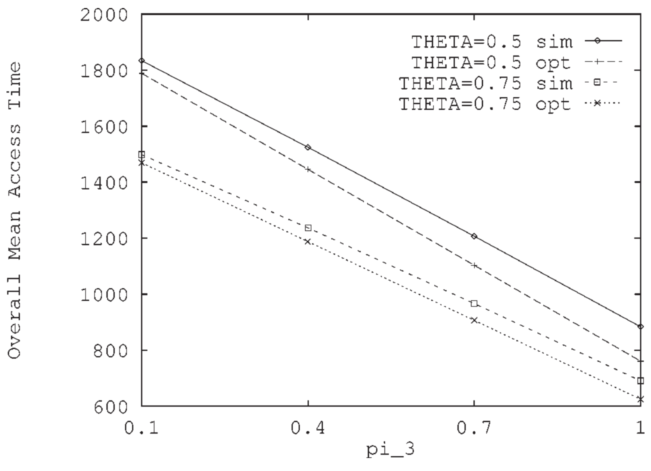
This section evaluates performance of the modified version of the single channel scheduling algorithm in the presence of uncorrectable errors, as explained in section 5. For the sake of illustration, we assume that uncorrectable errors occur according to a Poisson process with rate λ. Hence $E(l_i) = 1 - \mathrm{e}^{-\lambda l_i}$. Figures 5 and 6 plot *overall mean access time* in the presence of errors for different error rates (λ), and for decreasing and increasing length distributions, respectively. Again, in each of these figures, part (a) plots the simulation results and part (b) plots analytical lower bounds, for $\theta = 0, 1$ and 1.5. The lower bounds are obtained using equation (6) (substituting $E(l_i) = 1 - \mathrm{e}^{-\lambda l_i}$). Note that the results presented in section 7.1.1 correspond to the case when $\lambda = 0$. From the simulation results, observe that the single channel scheduling algorithm, modified to take errors into account, achieves performance close to optimal. Other researchers have not considered uncorrectable errors when designing schedules.

7.2. Performance evaluation of 2-channel broadcast algorithm

In this section, we evaluate performance of the 2-channel scheduling algorithm in section 6. For brevity, we only present results for the case when no transmission errors occur – similar results can be obtained when transmission errors do occur. Figures 7(a) and (b) plot the *overall mean access time* versus access skew coefficient θ for decreasing and increasing length distributions, respectively. The curves labeled "`ch1 sim`" and "`ch2 sim`" are the curves for $t_{\text{overall}(1)}$ and $t_{\text{overall}(2)}$, respectively, obtained from simulations. Recall that $t_{\text{overall}(i)}$ is the overall mean access time experienced by clients listening to first i channels. The curves labeled "`ch1 opt`" and "`ch2 opt`" plot t_{optimal} and $t_{\text{optimal}}/2$ – recall that t_{optimal}/i is a lower bound on $t_{\text{overall}(i)}$, where t_{optimal} is obtained from equation (3). The proposed 2-channel algorithm produces the same schedule irrespective of the values of π_1 and π_2. Therefore, the curves in figures 7(a) and (b) are applicable for all π distributions. Observe that, $t_{\text{overall}(i)}$ $(i = 1, 2)$ in these curves



(a) $\theta = 0, 0.2$



(b) $\theta = 0.5, 0.75$

Figure 8. Composite overall mean access time versus π_3, for random length distribution. The values of π_1 and π_2 are obtained as $\pi_1 = 2\pi_2 = \frac{2}{3}(1 - \pi_3)$. The curves labeled `sim` represent simulation results and `opt` represent analytical results. In (a), the curves shown are for access skew coefficient $\theta = 0$ and $\theta = 0.2$, whereas in (b), the curves shown are for $\theta = 0.5$ and $\theta = 0.75$.

is very close to t_{optimal}/i (the curves are almost overlapping). Therefore, it follows that the $t_{\text{composite\_overall}}$ (for any π distribution) will be very close to $t_{\text{composite\_optimal}}$ (see equations (7) and (8)).

7.3. Performance evaluation of 3-channel broadcast algorithm

Figures 8(a) and (b) show the performance of the 3-channel scheduling algorithm using the *random* length distribution. Similar results are obtained for the *increasing* and *decreasing* length distributions as well. For brevity, we only present results for the case when no transmission errors occur. As noted earlier in section 6, the values of ψ_{ij}, for $c \geqslant 3$, depend on π_i's. For $c = 3$, the values of ψ_{i2} and ψ_{i3} as a function of π's are given by equations (9) and (10).

In each figure in this section, the curves labeled `sim` plot the *composite overall mean access time* $t_{\text{composite\_overall}}$ obtained by simulations, and the curves labeled `opt` plot the lower bound $t_{\text{composite\_optimal}}$. These curves are plotted for different values of π_3 (horizontal axis) – π_1 and π_2 are defined as functions of π_3 as $\pi_1 = \frac{2}{3}(1-\pi_3)$ and $\pi_2 = \frac{1}{3}(1-\pi_3)$. Figure 8(a) plots the analytical and simulation curves for access skew coefficient, $\theta = 0$ and $\theta = 0.2$, whereas figure 8(b) plots the analytical and simulation curves for $\theta = 0.5$ and $\theta = 0.75$. In each of these figures, the curves labeled `sim` represent simulation results and those labeled `opt` represent analytical results. The analytical curves plot equation (8). The figures show that the performance of 3-channel scheduling algorithm is fairly close to optimal for some, but not all, values of access skew coefficient θ. The algorithm does not always perform well because of two reasons: (i) the bound $t_{\text{composite\_optimal}}$ is not tight for $c > 2$, and (ii) there may be room for improvement in our algorithm for $c = 3$.

8. Related work

The algorithms presented in this paper are based on an algorithm proposed previously for "packet fair queueing" [10]. As noted earlier, the problem of optimal broadcast scheduling is closely related to design of good packet fair queueing algorithms.

Some of the early work relevant to this paper was performed in the context of datacycle [12,22], and teletext and videotex [4,5,17,37,38] systems. 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* (e.g., [2,5,15,24,33,37]) and schemes attempting to reduce the *tuning time*, i.e., the time a client actively listens to the broadcast (e.g., [14,25,26,36]). In this paper, we only consider minimization of access time.

Ammar and Wong [5,37] have performed extensive research on broadcast scheduling and obtained many interesting results. An O(1) probabilistic approach for deciding which item to transmit next has been suggested previously [17,36,37]. 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 $l_i = 1$) [37]. On the other hand, with a logarithmic time-complexity, our single channel algorithm 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. Wong [37] also presents a cyclic scheduling algorithm that performs close to the optimal (the broadcast schedule needs to be generated *a priori*).

Chiueh [15] and Acharya et al. [2] present schemes that transmit the more frequently used items more often. However, they do not necessarily use optimal broadcast frequencies. Our schemes, on the other hand, tend to use optimal frequencies. (Optimal frequencies are inversely proportional to optimal spacing.)

Gondhalekar et al. [19] have looked at the problem of optimizing mean access time using indexing schemes, and shown that the problem is NP-complete under certain conditions. They also present fast heuristics to achieve a low access time using indexing. The scheduling schemes presented in this paper do not use indexing.

Several researchers, including Su and Tassiulas [32], Acharya et al. [2] and Stathatos et al. [31], have considered the possibility of caching information items at the client. With caching, a client need only wait for broadcast if the desired item is not in the cache. Our broadcasting schemes do not consider caching as yet.

9. Conclusions

This paper considers *asymmetric* environments where 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. This paper makes four contributions:

(i) Observes that the broadcast scheduling problem is related to packet fair queueing.

(ii) Presents a broadcast scheduling algorithm based on a packet fair queueing algorithm.

(iii) Modifies the above algorithm to take into account transmission errors.

(iv) Presents algorithms for scheduling broadcasts on multiple channels.

Simulation results suggest that the proposed algorithms perform well. These algorithms tend to result in near-optimal spacing between consecutive instances of a given item, achieving near-optimal performance.

Future work includes derivation of a better bound for $t_{\text{composite\_overall}}$, particularly, for $c \geqslant 3$. Also, this paper does not consider caching of information at a client. Proposed algorithms can be applied to a *pull*-based system by replacing p_i by the number of requests pending for item i, in our algorithms. We have not evaluated the proposed algorithms in the context of pull-based systems.

Acknowledgements

Thanks are due to P. Krishna for discussions on packet fair queueing and broadcast scheduling [29]. This paper appears in part in the proceeding of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), September 1997, Budapest. Research reported is supported in part by Texas Advanced Technology Program grant 009741-052-C and National Science Foundation grant MIP-9423735.

Appendix A. Average time complexity

The algorithm presented in section 4 is derived from a fair queueing algorithm by Bennett and Zhang [11]. The proof that our algorithm has average time-complexity (per iteration) of O($\log M$), follows directly from the fact that the algorithm in [11] has log-time complexity. However, to elaborate on how the logarithmic average time complexity can be achieved, we describe an implementation of the proposed algorithm. Bennett and Zhang apparently presented their implementation in [9]; however, we were unable to obtain a copy of [9] at the time of writing this paper. It is possible that their implementation of fair queueing is analogous to the implementation summarized below.

A binary heap [23] stores items in a tree form, such that the "key" for the item at the root of the heap is the smallest (or largest) of all items in the heap.

We maintain two binary heaps, H_B and H_C. Heap H_B uses B_i value as the *key*, and stores the item with the smallest B_i value, among all the items in H_B, at the root. Heap H_C uses C_i value as the *key*, and stores the item with the smallest C_i value, among all the items in H_C, at the root. The heap H_C implements set S in the algorithm in section 4.

Initially, H_B contains all M items, and H_C is empty. Due to the way the algorithm is implemented, each item belongs to exactly one of the two heaps, H_B and H_C, at any time.

In step 1 of the algorithm, set S can be determined by repeatedly removing item j at the root of heap H_B, such that $B_j \leqslant T$, and inserting it into H_C (note that after every removal or insertion of an item, a heap needs to be reheaped) – this process is completed when, for the item at the root of H_B, the B_j value is greater than T. Each insertion and removal of an item in a binary heap (including reheaping) requires O($\log M$) time [23]. Note that, in step 1, zero, one, or more items may be removed from H_B and added to H_C.

Steps 2 and 3 can be performed by removing the root item from H_C, in O($\log M$) time. Recall that heap H_C implements set S.

Assume that in step 3, item j is removed from H_C. In step 4, an instance of item j is broadcasted, and new values of B_j and C_j are calculated. At this time, the *next* instance of item j (with the new B_j and C_j values) is inserted into heap H_B.

The average time complexity can be determined by following the movement of each broadcast instance of an item. When a previous instance of item j is broadcast, the next instance of item j is inserted in H_B (in step 4 of the iteration that broadcasted the previous instance of item j, as noted above).

Subsequently, when time T becomes large enough such that $B_j \leqslant T$, the instance of j is removed from H_B and added to H_C (or set S), in step 1 of an iteration of the algorithm.

Eventually, when C_j for item j becomes smallest of all items in H_C, this instance of j is removed from H_C and transmitted.

Therefore, each broadcast instance of an item requires 4 heap operations, each operation requiring O($\log M$) time. Therefore, the average time complexity per iteration of the algorithm is O($\log M$).

Appendix B. Optimal values of stagger

B.1. Two channel broadcast

Figure 9 shows different instances of item i scheduled on two channels. The spacing on each of the channels is s_i. Every instance on channel 2 is staggered by an interval of ψ_{i2} from the corresponding instance on channel 1. Our interest is to determine the value of ψ_{i2} which will result in optimal *composite item mean access time*, denoted by t_i, as follows. Note that each composite t_i is being optimized independently – thus, all optimal t_i (or optimal stagger for all items) may not be achievable simultaneously.

The *item mean access time*, t_{i1}, for a client listening to channel 1, assuming that a request is equally likely to occur at any time in interval s_i, is clearly

$$t_{i1} = \frac{1}{2}s_i. \tag{B.1}$$

Note that the probability that a client makes a request during a sub-interval of length τ of an interval of length s_i is given by τ/s_i. Therefore, *item mean access time*, t_{i2}, for a client listening to both the channels can be obtained as

$$t_{i2} = \frac{1}{2}\frac{(s_i-\psi_{i2})^2}{s_i} + \frac{1}{2}\frac{\psi_{i2}^2}{s_i}. \tag{B.2}$$

Thus, the composite item mean access time can be obtained as

$$t_i = \pi_1 t_{i1} + \pi_2 t_{i2} \tag{B.3}$$

$$= \frac{1}{2}\pi_1 s_i + \frac{1}{2}\pi_2\frac{(s_i-\psi_{i2})^2}{s_i} + \frac{1}{2}\pi_2\frac{\psi_{i2}^2}{s_i}. \tag{B.4}$$

For the minimum value of t_i, we differentiate equation (B.4) with respect to ψ_{i2} and equate it to zero:

$$\frac{\mathrm{d}t_i}{\mathrm{d}\psi_{i2}} = -\pi_2\frac{(s_i-\psi_{i2})}{s_i} + \pi_2\frac{\psi_{i2}}{s_i} = 0.$$

Solving for ψ_{i2}, we get $\psi_{i2} = \frac{1}{2}s_i$.

Note that the value of ψ_{i2} for optimal *composite item mean access time* is independent of π_1 and π_2 for the two channel case. However, as can be seen in the next section, for $c = 3$, the value of ψ_{i2} for optimal *composite item mean access time* is a function of π_j's.

B.2. Three channels broadcast

Figure 3 shows the schedule for item i on three channels. Let the instances of item i on channel 2 be staggered by

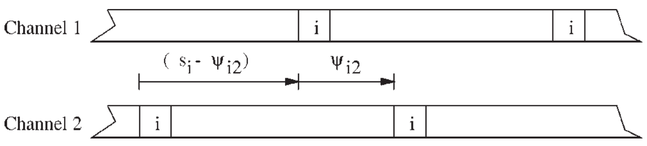


Figure 9. Schedule for item i on two channels. The instances of item i on channel 2 are staggered from channel 1 by an interval of ψ_{i2}. The value of ψ_{i2} should be $\frac{1}{2}s_i$ for the mean access time for item i to be minimum.

an interval of ψ_{i2} and on channel 3 be staggered by an interval of ψ_{i3} with respect to channel 1. A client may listen to channel 1 only, or to channels 1 and 2, or to all the three channels. The *item mean access times* for item i for a client listening to channel 1 and for a client listening to channel 1 and 2, denoted by t_{i1} and t_{i2}, and given by equations (B.1) and (B.2), respectively, are still valid, as the scheduling on first two channels in figure 3 is similar to the scheduling shown in figure 9. However, the *item mean access time* for item i for the client listening to all the three channels, denoted by t_{i3}, is given by

$$t_{i3} = \frac{1}{2}\frac{(s_i-\psi_{i3})^2}{s_i} + \frac{1}{2}\frac{\psi_{i2}^2}{s_i} + \frac{1}{2}\frac{(\psi_{i3}-\psi_{i2})^2}{s_i}. \tag{B.5}$$

From equations (B.1), (B.2) and (B.5), we get

$$\begin{aligned} t_i &= \pi_1 t_{i1} + \pi_2 t_{i2} + \pi_3 t_{i3} \\ &= \frac{1}{2}\pi_1 s_i + \frac{1}{2}\pi_2\frac{(s_i-\psi_{i2})^2}{s_i} + \frac{1}{2}\pi_2\frac{\psi_{i2}^2}{s_i} \\ &\quad + \frac{1}{2}\pi_3\frac{(s_i-\psi_{i3})^2}{s_i} + \frac{1}{2}\pi_3\frac{\psi_{i2}^2}{s_i} + \frac{1}{2}\pi_3\frac{(\psi_{i3}-\psi_{i2})^2}{s_i}. \end{aligned}$$

Again, for optimal t_i, differentiating the above equation with respect to ψ_{i2} and ψ_{i3}, we get

$$\frac{\partial t_i}{\partial\psi_{i2}} = -\pi_2\frac{(s_i-\psi_{i2})}{s_i} + \pi_2\frac{\psi_{i2}}{s_i} + \pi_3\frac{\psi_{i2}}{s_i} - \pi_3\frac{(\psi_{i3}-\psi_{i2})}{s_i} = 0, \tag{B.6}$$

$$\frac{\partial t_i}{\partial\psi_{i3}} = -\pi_3\frac{(s_i-\psi_{i3})}{s_i} + \pi_3\frac{(\psi_{i3}-\psi_{i2})}{s_i} = 0. \tag{B.7}$$

We assume that π_2 and π_3 are not both 0 – if both are 0, then the 3-channel problem reduces to the single channel broadcast problem. Solving equations (B.6) and (B.7), we get $\psi_{i2} = ((2\pi_2+\pi_3)/(4\pi_2+3\pi_3))s_i$ and $\psi_{i3} = ((3\pi_2+2\pi_3)/(4\pi_2+3\pi_3))s_i$. It can be verified that these values of ψ_{i2} and ψ_{i3} represent the point of minima, by applying appropriate checks to second derivatives of t_i [13].

The above proof can be generalized for $c > 3$ also.

References

[1] S. Acharya, R. Alonso, M. Franklin and S. Zdonik, Broadcast disks – data management for asymmetric communications environment, in: *Proc. of ACM SIGMOD Int. Conference on Management of Data* (May 1995) pp. 199–210.

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

[3] S. Acharya, M. Franklin and S. Zdonik, Prefetching from a broadcast disk, in: *Proc. of 12th International Conference on Data Engineering* (February 1996).

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

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

[6] 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).

[7] 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.

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

[9] J. Bennett and H. Zhang, Worst-case fair packet fair queueing algorithms, Technical Report, Computer Science, Carnegie Mellon University (1996).

[10] J.C.R. Bennett and H. Zhang, Wf2q: Worst-case fair weighted fair queueing, in: *Proc. of INFOCOM '96* (March 1996).

[11] J.C.R. Bennett and H. Zhang, Hierarchical packet fair queueing algorithms, in: *Proc. of ACM SIGCOMM '96* (1996) pp. 43–56.

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

[13] W.E. Boyce and R.C. DiPrima, *Calculus* (Wiley, New York, 1988).

[14] M.-S. Chen, P.S. Yu and K.-L. Wu, Indexed sequential data broadcasting in wireless mobile computing, in: *Proc. of International Conf. Distributed Computing Systems* (1997) pp. 124–131.

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

[16] 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: *Proc. of Data Engineering Conference* (April 1997).

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

[18] V.A. Gondhalekar, Scheduling periodic wireless data broadcast, M.S. thesis, The University of Texas at Austin, TX (December 1995).

[19] V. Gondhalekar, R. Jain and J. Werth, Scheduling on airdisks: Efficient access to personalized information services via periodic wireless data broadcast, in: *IEEE Int. Conf. Comm.* (June 1997).

[20] A. Gurijala and U. Pooch, Propagating updates in asymmetric channels (a position paper), in: *Proc. of 1st International Workshop on Satellite-based Information Services (WOSBIS)* (November 1996) pp. 53–59.

[21] S. Hameed, Scheduling information broadcast in asymmetric environment, M.S. thesis, Department of Computer Science, Texas A&M University (May 1997).

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

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

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

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

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

[27] 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).

[28] S. Keshav, On the efficient implementation of fair queueing, Journal of Internetworking: Research and Experience 2 (September 1991) 57–73.

[29] P. Krishna, Personal communication on packet fair queueing and broadcast scheduling (1996).

[30] M. Shreedhar and G. Varghese, Efficient fair queuing using deficit round robin, in: *Proc. of SIGCOMM '95*, Cambridge, MA (1995).

[31] K. Stathatos, N. Roussopoulos and J.S. Baras, Adaptive data broadcasting using air-cache, in: *Proc. of 1st International Workshop on Satellite-Based Information Services (WOSBIS)* (November 1996) pp. 30–37.

[32] C.-J. Su and L. Tassiulas, Novel information distribution methods to massive mobile user populations, Technical Report TR 97-46, ISR, University of Maryland (1997).

[33] N.H. Vaidya and S. Hameed, Data broadcast in asymmetric environments, in: *Proc. of 1st International Workshop on Satellite-Based Information Services (WOSBIS)* (November 1996) pp. 38–52.

[34] N.H. Vaidya and S. Hameed, Improved algorithms for scheduling data broadcast, Technical Report 96-029, Computer Science Department, Texas A&M University, College Station (December 1996).

[35] N.H. Vaidya and S. Hameed, Scheduling data broadcast in asymmetric communication environments, Wireless Networks 5 (1999) 171–182, this issue.

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

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

[38] J.W. Wong and M.H. Ammar, Analysis of broadcast delivery in a videotex system, IEEE Transactions on Computers 34 (September 1985) 863–866.

[39] Z. Zdonik, R. Alonso, M. Franklin and S. Acharya, Are disks in the air, 'just pie in the sky?', in: *Proc. of IEEE Workshop on Mobile Comp. System* (December 1994).

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.

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 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