Scheduling Algorithms for a Data Broadcast System: Minimizing Variance of the Response Time*

Shu Jiang Nitin H. Vaidya
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
Email: {jiangs,vaidya}@cs.tamu.edu

Technical Report 98-005

February 4, 1998

Abstract

Data broadcast has been suggested as a promising method of information dissemination in the communication environment with asymmetric characteristic. In such an environment, the information server cannot afford to serve the requests from a large population of users individually. Instead, the server uses a broadcast channel to deliver information to all users. A single transmission of a data item will satisfy all pending requests for that item.

However, the quality of service experienced by a single user turns out to be unpredictable and unstable. The response time of a request depends on the broadcast of the desired data item, which is scheduled by the server according to the overall demands for various data items. Therefore, the response time may vary in a large range, which may hinder a user subscribing to this service. We argue that, in addition to mean response time, the variance of response time should also be taken into account by the broadcast scheduler.

In this paper, we address the issue of variance optimization in regard to response time. Building on our previous research on mean response time optimization, we propose an algorithm which can minimize the variance of response time. Furthermore, we evaluate an algorithm that facilitates a trade-off between the mean and variance of response time. Numerical examples that illustrate the performances of our algorithms are also presented.

*This research is supported in part by Texas Advanced Technology Program under grants 009741-052-C and 010115-248-C.

Contents

1 Introduction

In a broadcast data delivery system, a server broadcasts data to a user community [3, 5, 6, 10]. The data is organized and transmitted in units called *items*. The items may be of different lengths and, certainly, of different demands. Some items are requested frequently; we call them *globally popular* items. On the other hand, some items may only be requested infrequently.

Since all the items contend for the use of broadcast channel, it is obvious that the allocation of bandwidth should favor the globally popular items because there are potentially (or actually) more pending requests for these items. In the past, several approaches for determining the actual broadcast schedule so as to reduce the global *mean* response time have been studied (e.g., [3, 5, 6, 10]). Unfortunately, this mean response time is not experienced by any particular user. Instead, it is experienced by a "virtual" user whose demand distribution is same as the demand distribution presented by the whole user population. In the real world, there may be a difference between the demand pattern of a particular user and the overall demand pattern. Therefore, the broadcast schedule, which is based on the overall demand pattern, may not be optimal for an individual user. This may lead to the response time experienced by a single user worse than the global average. Needless to say, the response time from the perspective of an individual user is of more concern than the global average from the perspective of the server[4].

To alleviate the impact of demand pattern difference between a single user and the user community on the system performance, the use of local cache has been suggested [2]. A user may prefetch and store in cache such items that are rarely broadcasted by the server (due to their unpopularity among the user community), but are needed by this particular user. Hence, the future requests to this kind of items by this particular user are satisfied by the cache. However, this method does not help when there is no cache in the user. In this case, the broadcast channel is the user's only source of information.

In this paper, we address this problem by adjusting the server's broadcast schedule. In essence, the process of scheduling broadcast is the process of allocating bandwidth resource among the items. As we mentioned before, minimizing the overall mean response time requires allocating more bandwidth to globally popular items, and the globally unpopular items end up being allocated less bandwidth, i.e., being scarcely broadcasted, which makes the values of the response time of all requests fall into a larger range. To say in mathematical language, the variance of response time becomes larger. From the perspective of the user, we believe that a low variance of response time is preferable (perhaps at the cost of a somewhat larger mean response time).

With this in mind, we argue that a new performance metric, variance of response time, should be added to evaluate the broadcast schedule. Though the minimal mean and minimal variance are both desirable, they cannot typically be achieved at the same time. Instead, we may try to find a trade-off between them.

The rest of the paper is organized as follows. In Section 2, we introduce our model of a *pure push-based*[2] data broadcast system and give the definition of variance of response time. Section 3 contains the analysis of a broadcast schedule and its relationship with mean and variance of response time. Our results are then used in Section 4 to propose scheduling algorithms which can minimize the mean response time or minimize the variance of response time or implement a balance between these two objectives. Section 5 discusses our simulations and some numerical results. We summarizes this paper in Section 6.

2 Model Description

Let M be the total number of available items in the system under consideration. These M items are stored in the database maintained by the server. We number these items from 1 to M and denote the length of item i with l_i. The time required to broadcast an item of unit length is referred to as a unit time. So, l_i is just the amount of time taken by item i when broadcasted.

We assume that there is only one broadcast channel in the system. The server can transmit an item only when the channel is idle. Consequently, the items are continuously broadcasted by the server and appear on the channel in sequence. A sequence of items on channel is called a *schedule*. The main task of a server is to find an appropriate schedule according to some criteria.

In a particular schedule, we assume that the first transmission of an item, say i, is given the sequence number 1. Subsequent transmissions of item i are given consecutive sequence numbers. For item i, the *spacing* $s_{ij}, j = 1, 2, \cdots$ is defined as the time between the beginning of the j-th transmission of item i and $j+1$-th transmission of item i in the schedule.

One performance measure of interest is the mean response time of requests which we denote by μ. Response time t of request r is defined as the elapsed time from when r is made until the desired item starts transmission. (We assume that the requests for item i arriving in the middle of a transmission of item i are not satisfied until the next transmission of item i.) Since we assume a *pure push-based* system, the server has no way to know the actual request stream generated by users.[1] Both the request arrival and the item required in each request are random events. So, t is a random variable and μ is actually the expected value of t.

$$\mu = E(t) \tag{1}$$

The mean response time has long been the primary performance metric which, as we said, only reflects the server's view of average quality of service (QOS) and what a single user experiences may be worse than expected. While a single user does need some globally popular items, it may also have "special" interests in some globally unpopular items which may make the waiting time untolerable. A better broadcast system should restrict the

[1]Algorithms presented in this paper can be easily adapted for a pull-based system where the server knows the number of requests pending for each item. In this case, the number of requests waiting for item i can be used as the current estimate of p_i. The algorithms presented here can then be used to achieve low variance in a pull-based system as well.

response time of request for any item into an acceptable range while maintaining the mean value as low as possible.

Therefore, we define a new performance metric, namely, the variance of response time t, denoted by σ^2.

$$\sigma^2 = E[(t-\mu)^2] \tag{2}$$

The physical meaning of σ^2 is that the response time of any request for any item is very likely to be within the range $(\mu - \sigma, \mu + \sigma)$. Low variance means that small difference of response time from the mean value can be expected. In the next section, we will show what kind of schedules produce low variance of response time.

An alternative approach may be specify an upper bound on response time of any request, and then to attempt minimization of the mean response time under this constraint. We do not consider this alternative in our current discussion.

3 Analysis

In the analysis to follow, we are concerned with deriving the properties of schedules which promise minimal mean response time and minimal variance of response time respectively.

Our analysis is based on an important assumption about the user request generation process, which is that, from the viewpoint of the server, a request is equally likely to be made at any time. Whereas this is not the case for a single user, i.e., both the time when a new request emerges and the item required in the new request may be related to previous requests, the consecutive requests from a large user population may be regarded as independent. As pointed out in [8], when the user population is large enough, we may assume that the aggregate request generation process is *Poisson* with constant rate.

Based on the assumption above, it has been observed that a "better" broadcast schedule has the Equal Spacing property which requires that the transmissions of each item on broadcast channel must be equally spaced. This observation can be explained by intuition. Suppose the transmissions of a particular item, say i, are not conducted constantly. Instead, there are "bursts" of item i in some time and there are long periods of sleep between the bursts. Since the requests for a particular item i arrive constantly and are only satisfied by the nearest transmission of item i, the requests coming in the sleeping period would wait a long time to be served and thus deteriorate the quality of service. In fact, the schedules produced by other schemes in [1], [8] all possess the "equal spacing" property, and we will consider only the schedules with this property in the following analysis.

In a particular schedule with Equal Spacing property, all transmissions of item i, $1 \leq i \leq M$, are equally spaced by some constant s_i and $s_{ij} = s_i, j = 1, 2, \cdots$. Now, a schedule can be specified by a vector, called *schedule vector*, $< s_1, s_2, \ldots, s_M >$ in which $s_i, 1 \leq i \leq M$ is the spacing for item i. We may regard the broadcast as the composition of M cyclic transmissions with each broadcasting an item in different frequency. In other words, this scenario is the generalization of Broadcast Disks scheme in [1]. While the latter applies only

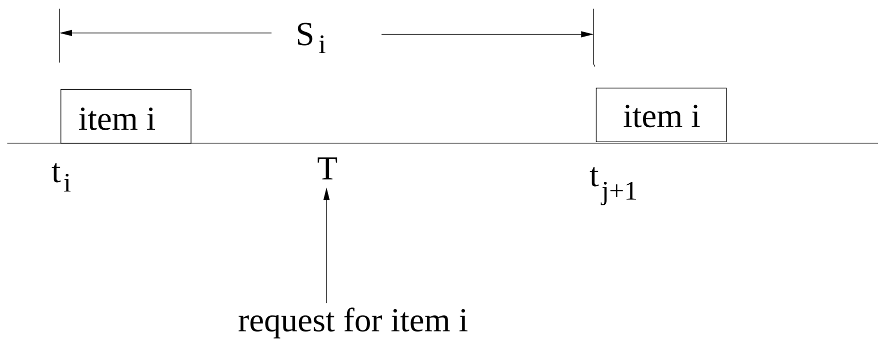

Figure 1: The relationship between t_j,T and t_{j+1}

to items of identical size, ours applies to items of differing sizes. Note that it is not always possible to construct a schedule strictly spacing the transmissions of each item as designated in a given schedule vector. But it is still of interest to us because it represents an ideal situation from which a theoretical lower bound can be achieved for reference. On the other hand, our simulation results show that the ideal results can be approached approximately in implementation, and even achieved exactly in some special configurations.

In the previous section, we stated that the response time t of any a request is a random variable whose distribution is determined by other two random variables: T, the issue time of the request, and I, the item required in the request. I is a discrete random variable taking integer values from 1 to M. The probability of I being i is called *demand probability* of item i.

$$p_i = Prob[\text{Item } i \text{ is requested in any request}], \qquad 1 \leq i \leq M$$

Obviously, it holds that $\sum_{i=1}^{M} p_i = 1$.

As for T, notice that if the request is for item i, only the schedule of item i is of significance to it. Assume that when the request arrives at time T, item i has been broadcasted for j times where j is zero or a positive integer.

Let t_j be the beginning time of j-th transmission of item i and t_{j+1} be that of $j+1$-th transmission which would be the transmission to serve the request (see Figure 1). The response time t is just the difference of T from t_{j+1}, i.e.,

$$t = t_{j+1} - T$$

As we mentioned before, an item is equally likely to be needed at any time (request arrival is governed by a Poisson process). So, T, as a random variable, is uniformly distributed in the range $(t_j, t_{j+1}]$, and hence the response time t for item i is also uniformly distributed in the range $(0, t_{j+1} - t_j]$. For a schedule with Equal Spacing property, $t_{j+1} - t_j$ is always a constant s_i (for item i), independent of j. Therefore, t is uniformly distributed over $(0, s_i]$ when $I = i$ (independent of j). Thus, the probability density function, say $q_i(t)$ for t given $I = i$ is:

$$q_i(t) = \begin{cases} \frac{1}{s_i} & , 0 < t \leq s_i \\ 0 & , \text{ otherwise} \end{cases}$$

Since t is a continuous random variable, cumulative distribution function for t is obtained as:

$$P[t \leq x | I = i] = F_i(x) = \int_{-\infty}^{x} q_i(t)dt \quad xreal$$

where $F_i(x)$ is the cumulative distribution function for t given that $I = i$.

Above is the conditional probability. Using the *Multiplication Rule* [7], we can derive the cumulative distribution function $F(x)$ for t.

$$P[t \leq x] = F(x) = \sum_{i=1}^{M}(Prob[I = i]Prob[t \leq x | I = i]) = \sum_{i=1}^{M}(p_i F_i(x))$$

Let g(t) be the probability density function of random variable t. It follows that,

$$g(t) \;\; = \;\; \sum_{i=1}^{M} p_i q_i(t) \tag{3}$$

Having the density function of t, it is not difficult to derive the expressions for μ, the expected value of t, and σ^2, the variance of t. The detailed derivation is given in Appendix I. Here we only report the results.

$$\mu = \frac{1}{2}\sum_{i=1}^{M} s_i p_i \tag{4}$$

and

$$\sigma^2 = \frac{1}{3}\sum_{i=1}^{M} p_i s_i^2 - (\frac{1}{2}\sum_{i=1}^{M} s_i p_i)^2 \tag{5}$$

or

$$\sigma^2 = \frac{1}{3}\sum_{i=1}^{M} p_i s_i^2 - \mu^2 \tag{6}$$

Note that expected (mean) response time and the variance of response time are only decided by the schedule vector (i.e., by s_i's). Using the expression for mean response time, [9] derives a law that must be obeyed by the schedule vector when the corresponding schedule minimized the mean response time μ. Specifically, to minimize the mean response time, spacing s_i of item i must be proportional to $\sqrt{l_i}$ and inversely proportional to $\sqrt{p_i}$, i.e.,

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

We prefer another form of the law, i.e

$$\frac{s_i^2 p_i}{l_i} = constant, \forall i, 1 \leq i \leq M \tag{7}$$

[9] has shown that there is only one schedule vector existent which satisfies this condition as well as exploiting the bandwidth resource to its maximum limit. The minimal value μ would take in this case is given as [9]

$$\mu_{optimal} = \frac{1}{2}\left(\sum_{i=1}^{M}\sqrt{p_i l_i}\right) \tag{8}$$

Minimizing Variance of Response Time

Similar to the above result, we found that a schedule vector must hold the following property in order to minimize the variance of response time:

$$\frac{p_i s_i^2}{l_i}\left(\frac{2}{3}s_i - \mu\right) = constant, \forall i, 1 \leq i \leq M \tag{9}$$

The proof of this result can be found in Appendix II.

The above two results provide valuable insight in the relationship between the schedule vector and the quality of service, as well as the theoretical basis for designing the scheduling algorithms. In the next section, we introduce a broadcast scheduling scheme which is based on these observations.

Notice that, in general, a schedule cannot achieve the equalities in Equation 7 and Equation 9 simultaneously. This can be proved easily. Suppose there is such a schedule. Since $\frac{s_i^2 p_i}{l_i}$ is a constant, say c_1, we may substitute the $\frac{s_i^2 p_i}{l_i}$ in the equality $\frac{p_i s_i^2}{l_i}(\frac{2}{3}s_i - \mu) = constant$, which should also be satisfied by this schedule. Now, we know that $\frac{2}{3}s_i - \mu$ is a constant, which cannot be true unless $s_i = constant, \forall i, i = 1, 2, \cdots, M$. s_i is a constant means that all items have same broadcast frequencies. This can only be achieved by a flat schedule. Furthermore, if s_i is a constant, $\frac{p_i}{l_i}$ must also be a constant to hold the equality $\frac{s_i^2 p_i}{l_i} = constant$. In another word, p_i must be proportional to l_i, which is a restrictive condition for most applications. Therefore, if a schedule makes the mean response time to be minimal, the variance of response time is usually not minimized and *vice versa.*

4 Algorithm

In our scheme, whenever the channel is idle, the server calls the proposed algorithm. The algorithm uses a *decision mechanism* to decide the item to be transmitted next[5]. The decision mechanism works like a "traffic police" in front of a crowd of vehicles contending the use of a single lane. The "police" has the responsibilities of coordinating the use of the road as well as keeping the road employed to full extent. In our system, the "road" is just the single broadcast channel and each item anxious to be broadcasted is a "vehicle". Actually we may find the role of "police" in many computer systems. For example, the scheduler in any multi-task operating system schedules the execution of processes with various strategies such as Round-Robin, First-Come-First-Serve, Small-Task-First, etc.

The decision mechanism in our algorithm uses a heuristic to help making decision. Let us look at an example. As we know, to implement a schedule which can make the mean response time very small, Equation 7 has to be maintained (at least approximately, if not exactly). Specifically, for each item, the square of its inter-transmission time multiplied by its demand probability and then divided by its length should be at least close to some constant, if not equal. However, the inter-transmission time increases with time if the item does not have the opportunity of being scheduled. So does the expression on the left side of Equation 7, whose change could be monitored by assigning each item with an indicator. Let Q be the current time and R_i be the time when item i was most recently transmitted.(If item i has never been broadcasted, R_i is initialized to -1.) A variable F_i can be defined as follows for item i.

$$F_i = (Q - R_i)^2 p_i / li \tag{10}$$

Notice that Q changes continually and R_i is updated whenever item i is transmitted. To keep the values of all F-indicators as close with each other as possible, as stated by the law, the item j with maximum F-indicator is a suitable target because its broadcast can update R_j to Q and thus bring F_j back to 0.

Algorithm for reducing mean response time :[9]

```
Step 1. For each item $i$, $1 \leq i \leq M$, update the value of F-indicator $F_i$.
Step 2. Determine maximum F-indicator over all items.
          Let $F_{max}$ denote the maximum value.
Step 3. Choose item $j$ such that $F_j = F_{max}$.
          If this equality holds for more than one item, choose any one of them arbitrarily.
Step 4. Broadcast item j.
Step 5. $R_j = Q$.
```

The definition of F-indicator in this algorithm is inspired by Equation 7. [9] has showed that it produces near-optimal result in regard to the mean response time. In the rest of this paper, we will refer to it as *Mean Optimal Algorithm.*

Reducing Variance of Response Time

We may use the above algorithm to reduce the variance of response time by replacing the definition of F-indicator with the following one, motivated by Equations 9 and 4.

$$F_i = \frac{p_i(Q - R_i)^2}{l_i} \Big(\frac{2}{3}(Q - R_i) - \frac{1}{2} \sum_{i=1}^{M} p_i (Q - R_i) \Big), 1 \leq i \leq M \tag{11}$$

. With this definition, we are now trying to maintain the equality in Equation 9 to the extent possible. We will refer to the new algorithm as *Variance Optimal Algorithm* in next section. Note that the name *Variance Optimal* may be a misnomer, as the algorithm is not *proved* to achieve near-optimal variance (as we do not know a tight lower bound on variance).

As we pointed out in last section, minimal mean and minimal variance are usually impossible to achieve simultaneously. When mean response time is reduced to minimal, the variance of response time may climb to a height which is perceivable by users. But if we turn to minimize the variance, mean response time is sure to be large. Both situations are not welcomed by users. The solution is to find an intermediate state between the two extreme cases, i.e., trade-off mean with variance of response time.

We notice that the expressions of calculating F-indicators in Mean Optimal Algorithm and Variance Optimal Algorithm both are polynomials of $Q - R_i$. A quadratic polynomial is used in Mean Optimal Algorithm and a cubic polynomial in Variance Optimal Algorithm. Basically, every time the decision mechanism executes, it evaluates each item according to demand probability, length and the time since its last transmission, and chooses the most suitable one. Even for an extremely unpopular item, when it has long been neglected, the time factor becomes so important that the effects of demand probability and length can be offset and the opportunity of broadcasting is given to it. In Mean Optimal Algorithm, the time of each item since last transmission is squared before evaluation. However, it is cubed in Variance Optimal Algorithm, which means that the time factor plays a more important role and is of advantage to globally unpopular items. We believe that a trade-off exists between the two styles of using the time factor in evaluating the eligibilities of items. More specifically, we define the F-indicator of item i as the polynomial of $Q - R_i$ which is of degree α and α is between 2 and 3. The following is the new expression of calculating F-indicator.

$$F_i = (Q - R_i)^{\alpha} p_i / li, 2 \leq \alpha \leq 3 \tag{12}$$

We call the corresponding algorithm the α-algorithm. When $\alpha = 2$, it becomes the Mean Optimal Algorithm. The α-algorithm was also evaluated by Su and Tassiulas [8]. They simulated the α-algorithm for various values of α, and impirically showed that $\alpha = 2$ minimizes the mean response time. We obtained the same result analytically in our prior work. Su and Tassiulas, however, did not consider the impact of varying α on the variance of the response time. When α is picked close to 3, it is expected to produce a schedule which can make the variance of response time small. Although we cannot provide any analytical evidence for this claim, the simulation results in next section support it indeed.

Appendix III derives expressions for the lower bounds on mean and variance of response time achieved using the α-algorithm. The lower bounds for one set of length and demand probability distributions are plotted in Figure 2.

5 Performance Evaluation

In this section, we present some numerical results from our simulation of a broadcast data delivery system. The server uses various algorithms we presented above to do scheduling. The request generation by users and request service process are also simulated. The data of response time measured from the simulation is analyzed. From the results, all algorithms can be seen clearly to be performing as expected.

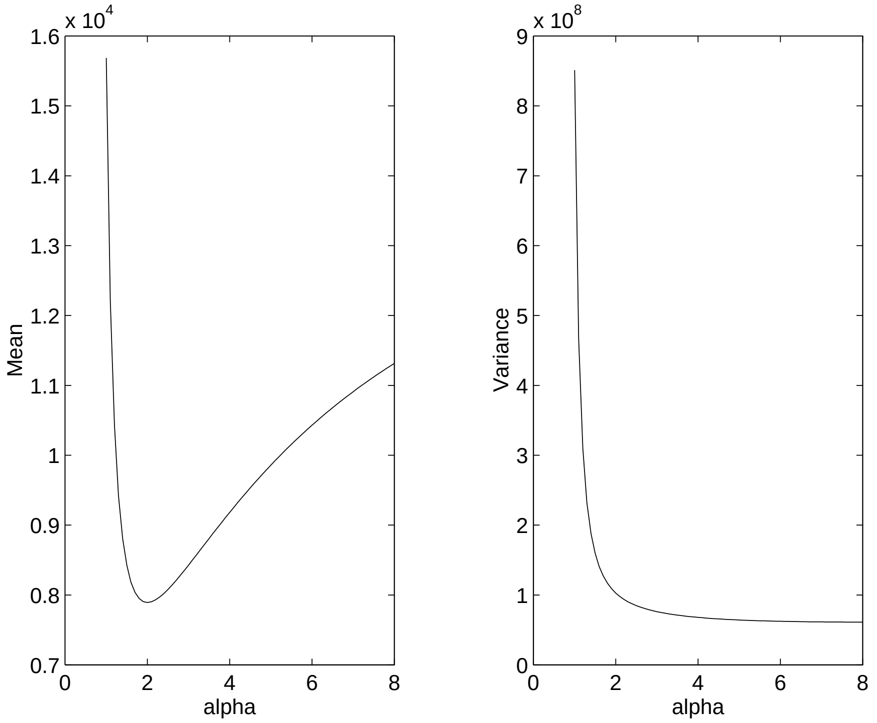

Figure 2: The lower bounds on mean and variance of response time when α-algorithm is used as scheduling algorithm and other system parameter settings are: $M = 250, \theta = 0.75, Increasing \quad Length \quad distribution($ θ and length distribution are defined in Section 5).

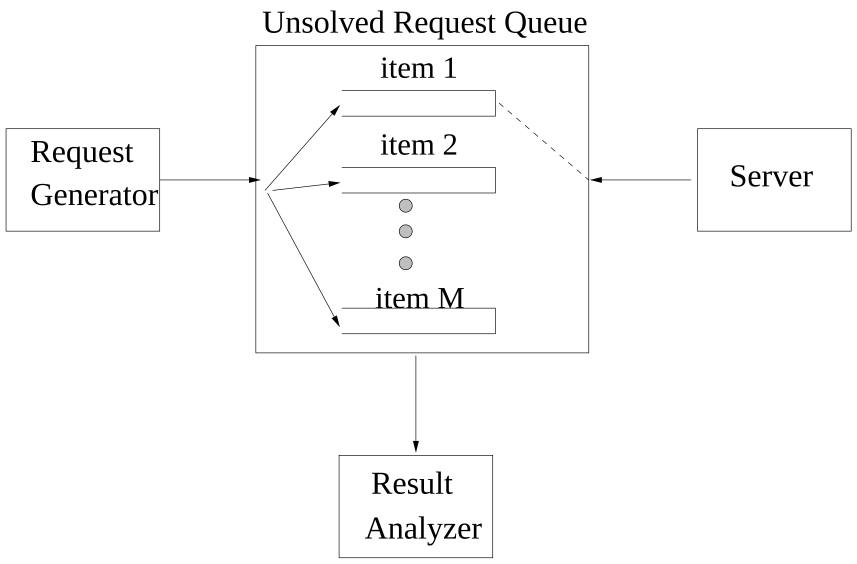

Figure 3: Architecture of Simulator

5.1 Simulation Model

5.1.1 Simulator

We developed an event-driven simulator. The architecture of the simulator is depicted in Figure 3. In real world, each user falls into a loop of making a request for an item, waiting until the request is served by broadcast, computing or thinking for a random time and making a new request again. However, the server only "sees" a stream of requests flowing out of the user community. In the finite user population case, the flow rate of request stream will drop as more users enter the waiting stage, and rebound if the broadcast of an item happens to resolve a large number of pending requests and free the users from sleeping. But with our assumption of large user population, the rate of request arrivals remains constant. We may just use a *Request Generator*, which produces a request series according to a *Poisson* process, to simulate the collective behavior of requests made by all users in the group. In our simulations, the request arrival rate is 2 per time unit.

All the newly generated requests enter into a data structure called *Unsolved Request Queue*, which contains M queues each having infinite capacity and holding all requests for an item. When an item is scheduled to broadcast, the corresponding queue is checked and all requests in it are satisfied together regardless of when these requests arrived, and the queue becomes empty. In the process of resolving requests, the response time of each request is measured and sent to a module called *Result Analyzer* as well as other useful information such as the item requested. The *Result Analyzer* stores, analyzes the experimental data and posts the results in windows immediately. Then the simulation process can be monitored and controlled easily.

The function of the server is implemented in the *Server* module, which accepts the demand probability distribution information of all items as the parameter and executes one of the algorithms we introduced in last section. The result of every execution of the algorithm is the number of item to be transmitted right now, which is routed to the *Unsolved Request Queue* to resolve the pending requests.

Some auxiliary modules are not included in the figure. For example, the *Configuration* module provides the interface for us to change system parameters such as M, the total number of items, α, the trade-off coefficient in our α-algorithm, etc.

5.1.2 Demand Probability Distribution Of Items

In our simulation, the demand probabilities of all items follow Zipf distribution, with item 1 being the most frequently requested, and item M being the least frequently requested. The Zipf distribution may be expressed as follows:

$$p_i = c\left(\frac{1}{i}\right)^{\theta}, 1 \le i \le M$$

where $c = \frac{1}{\sum_{i=1}^{M}(\frac{1}{i})^{\theta}}$ is a normalizing factor, and θ is a parameter named *access skew coefficient*. When $\theta = 0$, Zipf distribution reduces to a uniform distribution with each item equally likely to be requested. However, the distribution becomes increasingly "skewed" as θ increases(that is, the difference among items with respect to the degree of popularity becomes more significant.

5.1.3 Length Distribution Of Items

As of length distribution, the following three special cases are considered in our simulation:

1. **Equal length case**:
   All items are equally sized and the size is 1, without loss of generality.

2. **Unequal length case**:

   - Increasing Length Distribution:

     $$l_i = l_{min} + \frac{(i-1)(l_{max} - l_{min} + 1)}{M}, i = 1, 2, \cdots, M \text{ with } l_{min} = 1 \text{ and } l_{max} = 250$$

     In this case, the most popular item, i.e. item 1, is the longest item.

   - Decreasing Length Distribution:

     $$l_i = l_{max} - \frac{(i-1)(l_{max} - l_{min} + 1)}{M}, i = 1, 2, \cdots, M \text{ with } l_{min} = 1 \text{ and } l_{max} = 250$$

     In this case, the most popular item, i.e. item 1, is the shortest item.

M	250
p_i	$\theta = 0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5$
l_i	Equal, Increasing, Decreasing
α	2.2, 2.6, 3

Table 1: Parameter Settings

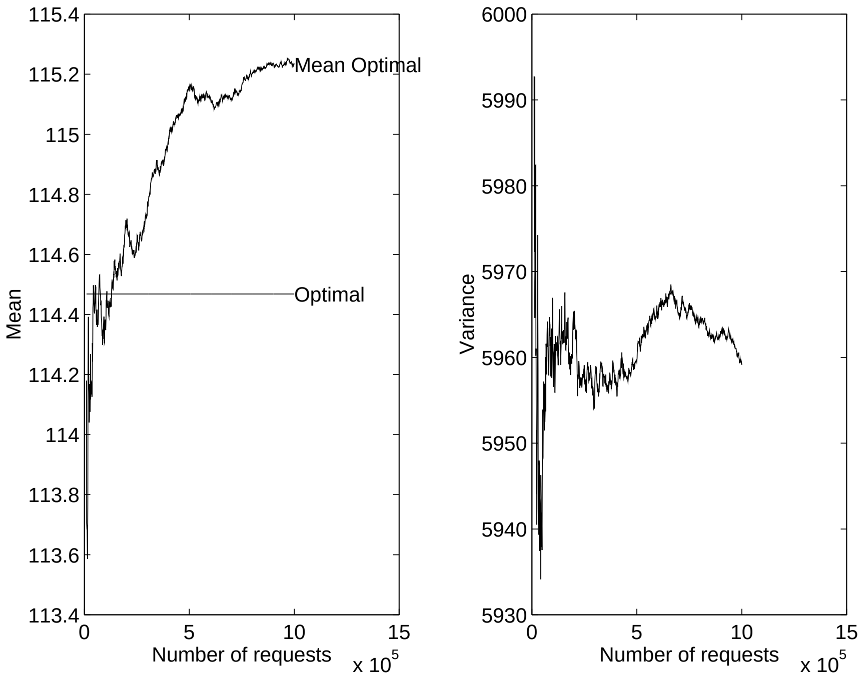

Figure 4: The dynamic changes of μ and σ^2 during a simulation

5.2 Numerical Results

Table 1 shows the parameter settings for our simulations. We conducted a number of experiments under different combinations of the parameter settings. The primary performance metrics are mean and variance of response time. We explored the trade-offs between the two performance goals. In addition to α-algorithm, we also measured the performance when Mean Optimal Algorithm and Variance Optimal Algorithm are used by the server respectively.

The mean and variance results with respect to response time were obtained once 1 million requests are served. During the simulation, both the mean and variance of response time keep changing as the number of requests served increases, starting from 0. Figure 4 shows how these two metrics change in one of our experiments. At the beginning of the simulation, a big change of both mean and variance can be identified. This phenomenon may be explained

	Mean Optimal Alg	α-algorithm			Variance Optimal Alg
		2.2	2.6	3	
Mean	125.005	125.005	125.005	125.005	125.005
Variance	5214.04	5214.04	5214.04	5214.04	5214.04

Table 2: Performance of different algorithms in Experiment 1: Uniform Demand, Equal Length

as caused by bad settings of initial values. Remember that in each of our algorithms, the last transmission time of any item plays a crucial role in the decision making process. When the server starts working, the last transmission time of any item is initialized to -1, which implies that in the "race" of F-indicators, all items start from a same starting line despite the fact that these items should be treated discriminately. Therefore, it can be imagined that the scheduling results worked out at that time is far away from the best, and the drastic changes in regard to the mean and variance of response time can be anticipated. After the system is running for a while, both mean and variance fluctuate only in a small range of values. We may claim that the system has entered into steady state and it is providing the best performance it could. We find that in all our experiments, the system had reached steady state when 1 million requests have been served (note that request arrival rate is 2 per time unit). In order to eliminate the "warm-up" effects and make the experimental data reflect the situation in steady state more accurately, we start measurements only after the server has broadcasted 5000 items. From then on, the served requests are counted until the number reaches 1 million.

In Figure 4, note that, the estimate of mean response time is initially *smaller* that the optimal. This may seem counter-intuitive. The reason for this phenomenon is that, the estimate is calculated based on requests that have been *served.* Requests waiting to be served are not taken into account. Initially, the more popular items tend to be broadcasted first, therefore, the initial estimate of mean response time, reflects the response time for the more popular items. Over time, all items are broadcasted multiple time, and the estimate of mean approaches its real value.

5.2.1 Experiment 1: Uniform Demand, Equal Length

In the first experiment, we assumed that all the 250 items have length 1, and the value of skew coefficient θ is 0, i.e., each item has the same chance of being required. The data in Table 2 shows that all algorithms produce same results.

When all items are of same size and accessed equally likely by users, the server tends to broadcast all items alternatively and actually uses the Round-Robin strategy because the demand probabilities and lengths of items are not considerable factors in the process of making scheduling decisions. By analyzing the trace of simulation, we found that all five algorithms produce same schedules, i.e flat schedule. Needless to say, same mean and

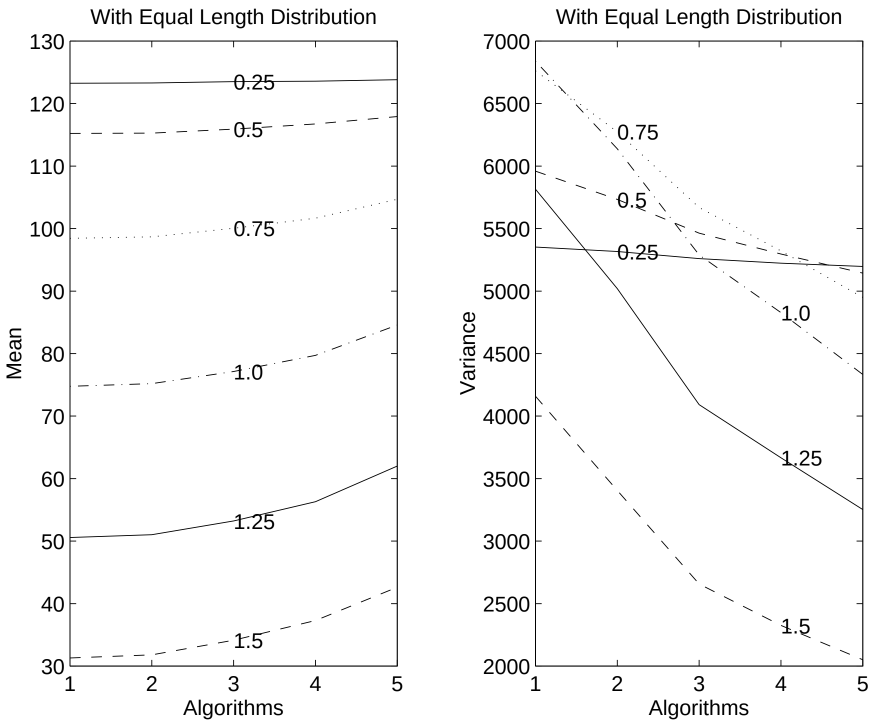

Figure 5: Performance of different algorithms in Experiment 2: Non-uniform Demand, Equal Length (1:Mean Optimal Algorithm, 2:α-algorithm with $\alpha = 2.2$, 3:α-algorithm with $\alpha = 2.6$, 4:α-algorithm with $\alpha = 3$, 5:Variance Optimal Algorithm)

variance results can be expected. In fact, the quality of service cannot be better. As we have proved in Section 3, when demand probability of each item happens to be proportional to its length, which is just the case in Experiment 1, the flat schedule is the best schedule in the sense that the minimal mean response time and the minimal variance of response time can be achieved simultaneously.

5.2.2 Experiment 2: Non-uniform Demand, Equal Length

In this experiment, we kept our Equal Length assumption about length distribution of all items. But the demand distribution of all items was no longer uniform. Instead, the skew coefficient θ changes from 0.25 to 1.5 gradually. We measured the performance of all algorithms in different cases. The results are shown in Figure 5, Table 3 and Table 4. In the figure whose y-axis is labeled "Mean", a curve demonstrates the change of means caused by using different scheduling algorithms in a particular environment. The number marked on the curve is the value of skew coefficient θ in that situation. From left to right along the x-axis, the algorithms we used in our experiment are numbered consecutively from 1 to 5 and they are Mean Optimal Algorithm, α-algorithm with $\alpha = 2.2, 2.6, 3$ and Variance Optimal Algorithm, respectively.

The key observations are as follows. The lowest mean response time is achieved when server is using the Mean Optimal Algorithm, and the lowest variance of response time when using Variance Optimal Algorithm. Furthermore, the α-algorithms indeed implement the balance between mean and variance of response time. When α changes from 2.2 to 2.6 and then to 3, the measured mean response time is observed to increase gradually while variance is dropping at the same time. This is consistent with the conjecture mentioned above.

However, the effectiveness of α-algorithm and Variance Optimal Algorithm in reducing variance of response time is challenged when the skew in user demands for items is small. When $\theta = 0.25$, mean and variance do not show any significant change (when the algorithm is varied). Actually, the mean response time results produced by 5 algorithms are so close with each other that the difference between the maximum value and the minimal value is less than 1 time unit and all are very close to the theoretically minimal value.

When θ increases but is still less than 1.25, the skew in user demands becomes a little large but not too large. For instance, when $\theta = 0.75$ and the number of items is 250, about half user requests are for top 33 items and the remaining 217 items take the burden of serving the other half requests. However, to reduce the overall mean response time, most bandwidth is given to a few items by Mean Optimal Algorithm. The poor service for the "not most popular" items worsens the overall quality of service which is reflected by the high variances of response time in these cases. Both α-algorithms and Variance Optimal Algorithm greatly improve the situation. From the curves, the reduction of variance brought by those algorithms is conspicuous, and the unavoidable increase of mean is not very significant.

When θ further increases to 1.5, we may find that the performance of Mean Optimal Algorithm in regard to variance is not very bad. This phenomenon looks strange at first. In fact, when the skew in user demands becomes extremely large, the percentage of requests attracted by a few hot items becomes very large. In the same example of 250 items, if $\theta = 1.5$, top 33 items are demanded by 91% user requests. To serve them well means to serve all well. That is why Mean Optimal Algorithm does not add the variance very much while still providing the best mean response time. On the other hand, even though α-algorithms and Variance Optimal Algorithm reduce the variance of response time drastically as expected, they make a large sacrifice with respect to mean response time. When most requests not only from the whole user community but also from a particular user are for a few items, a little increase of response time may give rise to a large increase of overall user waiting time.

In summary, α-algorithms and Variance Optimal Algorithm perform best in the situation with medium-skewed demand distribution, and when user demands are lightly skewed or severely skewed, Mean Optimal Algorithm is still a good alternative.

5.2.3 Experiment 3: Non-uniform Demand, Unequal Length

In this experiment, Increasing Length distribution and Decreasing Length distribution were assumed respectively. The results for increasing lengths are shown in Figure 6, Table 5 and Table 6, those for decreasing lengths in Figure 7, Table 7 and Table 8. Also, the skew coefficient θ changes from 0.25 to 1.5 simulating various situations about how user demands

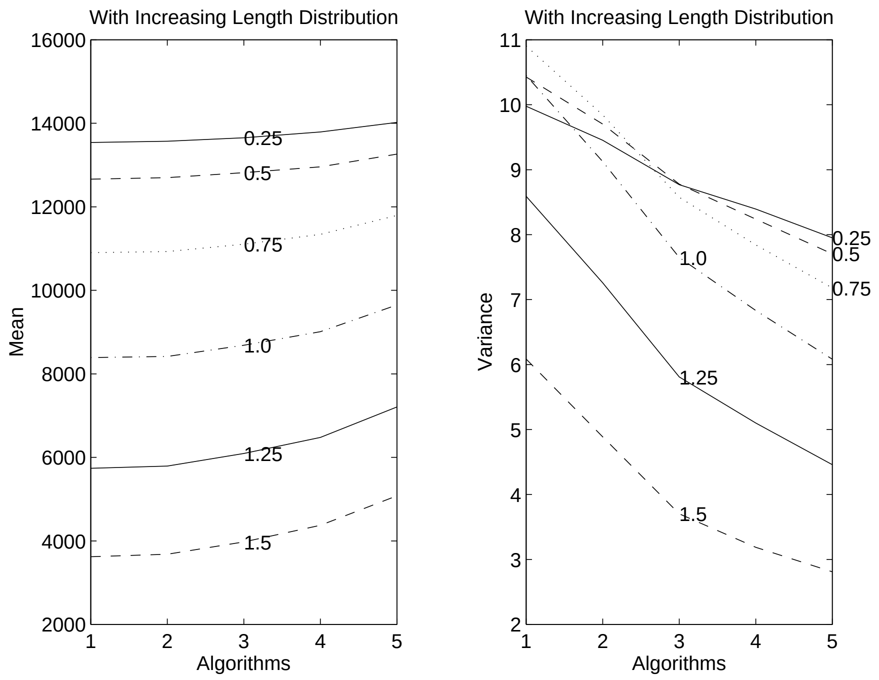

Figure 6: Performance of different algorithms in Experiment 3: Non-uniform Demand, Increasing Length (1:Mean Optimal Algorithm, 2:α-algorithm with $\alpha = 2.2$, 3:α-algorithm with $\alpha = 2.6$, 4:α-algorithm with $\alpha = 3$, 5:Variance Optimal Algorithm)

are distributed among the items.

From the curves, the performance of these scheduling algorithms seems to be insensitive to the length distribution of items. As in Experiment 2, in which all items have equal sizes, the α-algorithms and Variance Optimal Algorithm reduce the variance of response time in all cases without exception. The parameter α in an α-algorithm takes the effect of adjusting the balance between minimal mean and minimal variance. When α is closer to 2, the power of minimizing the mean becomes stronger and the α-algorithm performs more similar to the Mean Optimal Algorithm. When α is closer to 3, the factor of minimizing variance weighs more heavily in the schedule making process.

Also, as in Experiment 2, the results show that the system environment with user demands for items medium-skewed is the most suitable one for α-algorithm and Variance Optimal Algorithm to work, where the variance of response time drops greatly but mean response time rises only a little.

5.2.4 Equal Spacing property

In this section, we examine the actual inter-transmission time of items in simulation and see if the Equal Spacing condition can be achieved (or approximated) by our algorithms. Figure 8 shows how the broadcast spacing of item 1 changes in its first 500 transmissions. Along the x-axis, the transmissions of item 1 are numbered from 1 to 500. The y-coordinate of a point

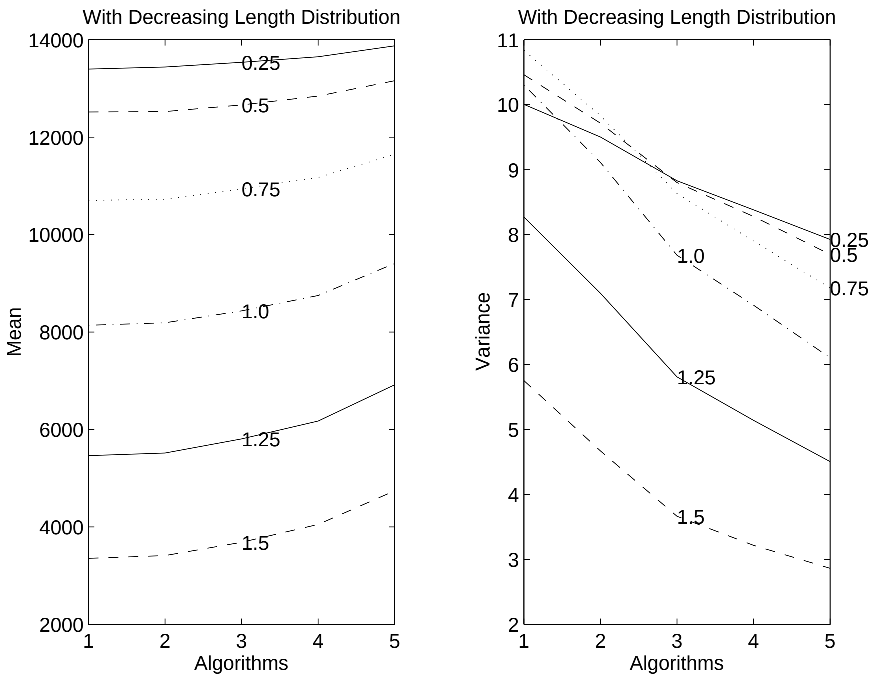

Figure 7: Performance of different algorithms in Experiment 3: Non-uniform Demand, Decreasing Length (1:Mean Optimal Algorithm, 2:α-algorithm with $\alpha = 2.2$, 3:α-algorithm with $\alpha = 2.6$, 4:α-algorithm with $\alpha = 3$, 5:Variance Optimal Algorithm)

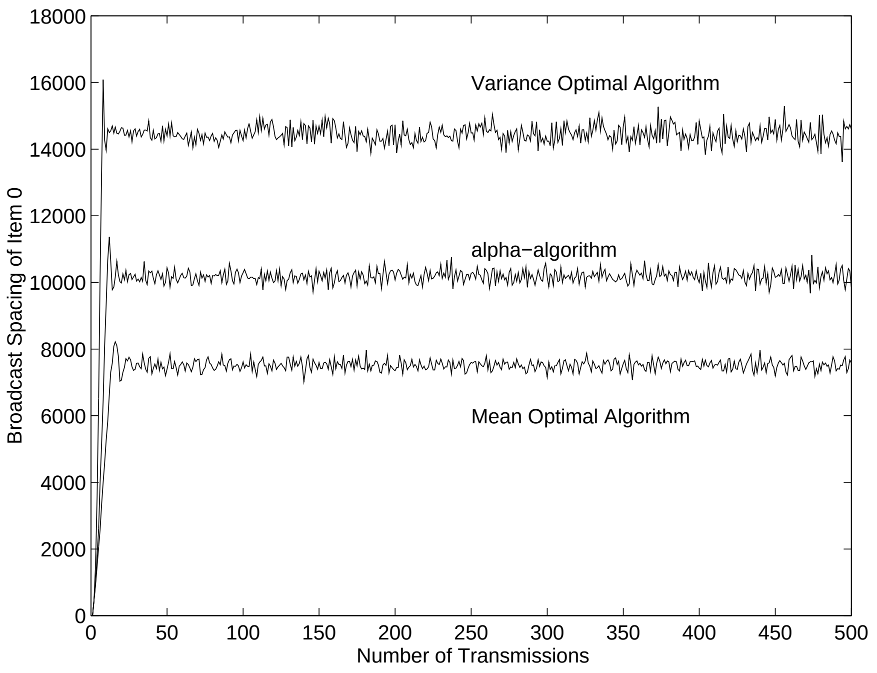

Figure 8: The broadcast spacing of item 1

on curve with x-coordinate i is the time interval between i-th transmission of item 1 and its (i-1)-th transmission, i.e spaing s_{0i}. The curve labeled "Mean Optimal Algorithm" plots the spacing data measured in a simulation with the server using Mean Optimal Algorithm as its scheduling algorithm. Keeping all other system parameters unchanged and replacing the scheduling algorithm with Variance Optimal Algorithm at first and α-algorithm with $\alpha = 2.6$ then, we conducted the simulation again. The other two curves illustrate the results in the two simulations respectively.

Look at the curve labeled "Mean Optimal Algorithm". The specific values of spacings are not important. What's important is the fact that the broadcast spacings of item 1 are bounded and the curve turns out to fluctuate around a horizontal line, which implies that the spacings are nearly same with each other and item 1 is broadcasted almost constantly spaced. Same conclusion may be drawn from other two curves too. As a matter of fact, the broadcasts of all other items demonstrate this fact as well, which indicates that our scheduling algorithms do create schedules with Equal Spacing property approximately true. The relative position of the three curves in Figure 8 is also very interesting. Compared with the situation when Mean Optimal Algorithm is used, item 1, the most globally popular item, occupies less bandwidth with Variance Optimal Algorithm, which means that the broadcast frequency of item 1 is lower and the spacing is larger as illustrated. α-algorithm with $\alpha = 2.6$ is the trade-off of those two extremes. So, it is quite reasonable that the curve of α-algorithm stays between the other two.

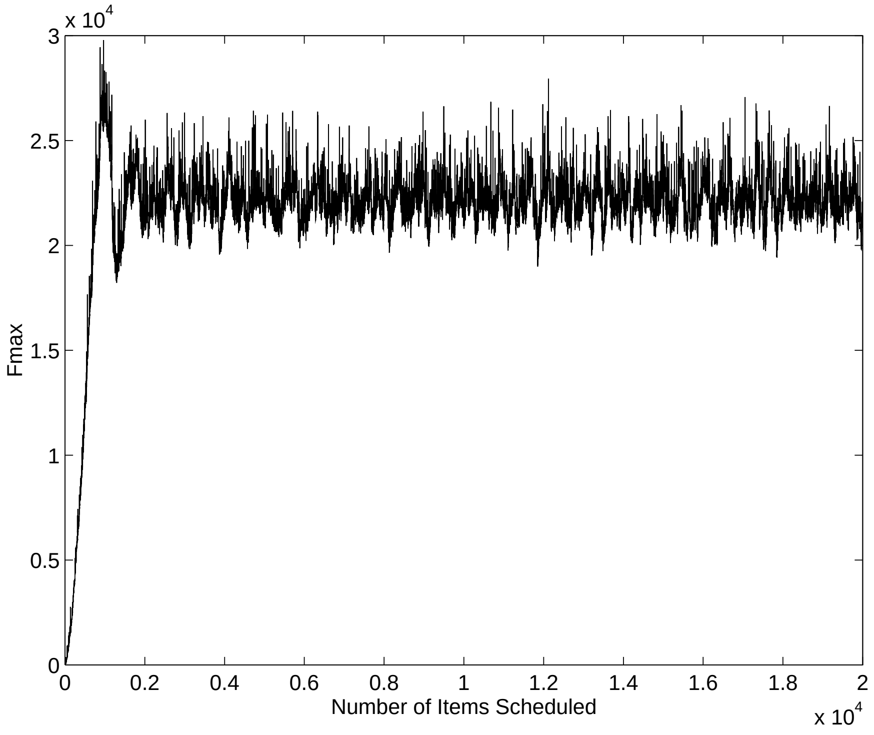

Figure 9: The change of F_{max} (Mean Optimal Algorithm)

5.2.5 F_{max} in simulation

In simulation, we tracked the change of F_{max} in each algorithm. The performance of each algorithm depends greatly on whether the F-indicators for all items are maintained close to each other. Figures 9, 10 and 11 plot the values of F_{max} against the number of items scheduled for three scheduling algorithms. These simulations use the Increasing Length distribution and skew coefficient $\theta = 0.75$.

From Figure 9, where Mean Optimal Algorithm is used as scheduling algorithm, it can be clearly seen that F_{max} fluctuates around a constant. Same phenomenons are also observed in the other two figures where α-algorithm with $\alpha = 2.6$ and Variance Optimal Algorithm are used respectively. Therefore, it appears that each algorithm maintains the stability of maximal F-indicator reasonably well, which is an important condition to make the algorithm perform near-optimally.

In theory, if the maximal F-indicators (i.e., F_{max}) can always be kept constant, optimal performance (with respect to mean or variance) can be expected. Equation 8 provides the minimal mean response time which can be achieved by Mean Optimal Algorithm in theory. For the α-algorithm, the theoretical values for optimal mean and variance of response time can also be obtained (as shown in Appendix III). Unfortunately, as illustrated in Figure 10, the equation $F_i = \textit{constant}$ was not held strictly by the algorithm. So, the real performance is bound to be a little worse than the theoretical optimal.

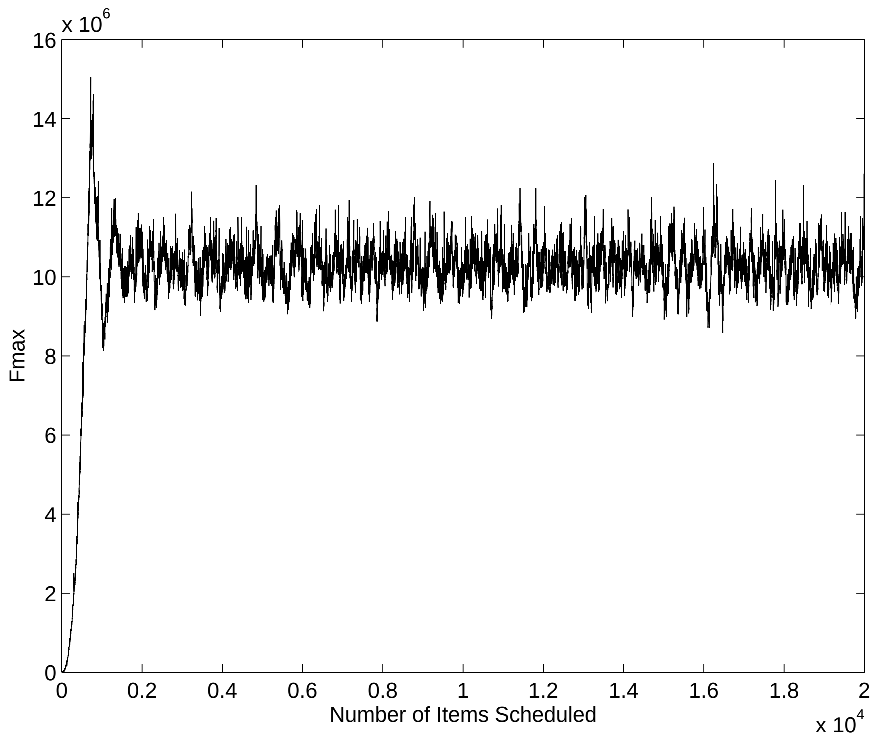

Figure 10: The change of F_{max} (α-algorithm with $\alpha = 2.6$)

6 Conclusion

In this paper, we address the problem of adding user perspective to the server scheduling process. We argue that from a user's point of view, the variance of response time is also important. As bad experiences usually stay longer than good ones in a person's memory, variance of response time affects a user's impression about the quality of service a system can provide. In the so-called *pure push-based* data broadcast system where there is no direct channel for users to send requests explicitly, the server may reduce the variance of response time by making appropriate broadcast schedules. In particular, we found the property a schedule must possess in order to achieve minimal variance of response time. Based on our finding, a scheduling algorithm was proposed for a server to construct such a schedule. While minimal mean and minimal variance are generally impossible to be achieved simultaneously, a trade-off does exist. We evaluated an algorithm that can achieve such a trade-off.

The contributions of this paper are in two aspects. First, we introduced the new performance metric, i.e., variance of response time. Second, we evaluated scheduling algorithms which can minimize the variance, or trade the variance of response time with the mean response time.

The evaluation presented in this report assumed a push-based system. Our algorithms can be easily adapted to achieve low variance in *pull-based* systems. In this case, the number of requests waiting for a particular item can be used in place of *demand probability* of the

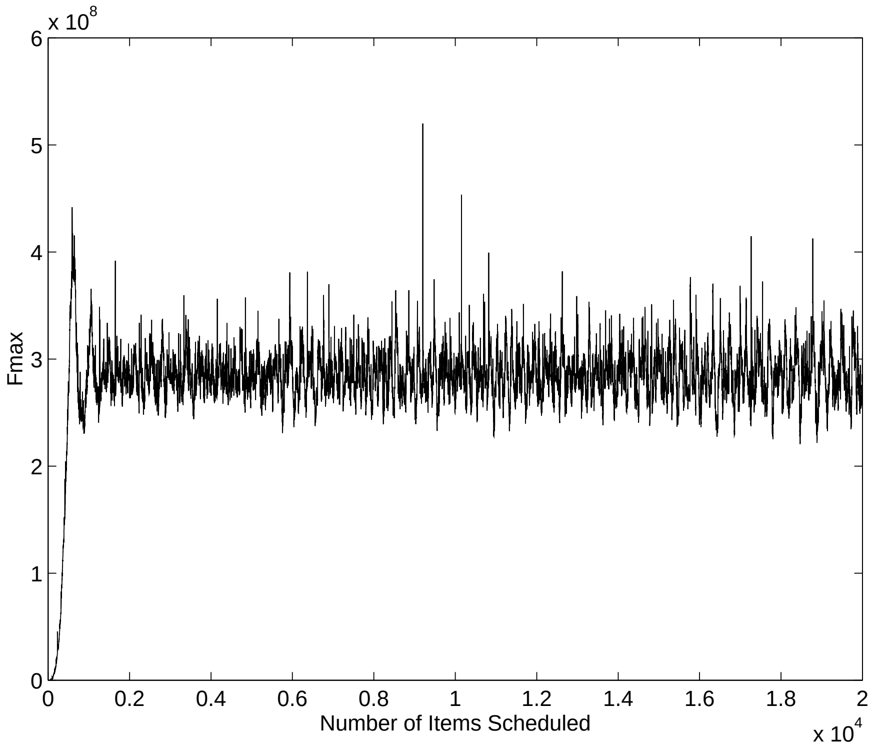

Figure 11: The change of F_{max} (Variance Optimal Algorithm)

	Mean Optimal Alg	α-algorithm			Variance Optimal Alg
		2.2	2.6	3	
$\theta = 0.25$	123.255	123.305	123.522	123.596	123.832
$\theta = 0.5$	115.233	115.278	115.912	116.731	117.931
$\theta = 0.75$	98.4467	98.6771	100.047	101.67	104.692
$\theta = 1.0$	74.7596	75.1937	77.1408	79.7116	84.5711
$\theta = 1.25$	50.5683	51.0305	53.2272	56.3029	62.0131
$\theta = 1.5$	31.3046	31.7905	34.133	37.2893	42.6269

Table 3: Mean response time results measured in Experiment 2: Non-uniform Demand, Equal Length

	Mean Optimal Alg	α-algorithm			Variance Optimal Alg
		2.2	2.6	3	
$\theta = 0.25$	5352.67	5316.94	5260.15	5223.47	5196.8
$\theta = 0.5$	5959.11	5733.29	5463.06	5296.69	5143.41
$\theta = 0.75$	6762.51	6274.52	5667.94	5321.34	4950.00
$\theta = 1.0$	6838.76	6137.74	5289.61	4827.42	4332.60
$\theta = 1.25$	5815.16	5019.19	4091.45	3666.95	3252.81
$\theta = 1.5$	4159.51	3404.48	2657.28	2325.43	2051.40

Table 4: Variance of response time results measured in Experiment 2: Non-uniform Demand, Equal Length

item.

One open problem that needs to be addressed is derivation of a good lower bound on the achievable variance of response time. Another issue to be studied is the impact of local cache at a client on the client's variance of response time (or, how to use a local cache to reduce or trade variance with mean response time). Transmission errors were not considered in this work. Such errors tend to adversely affect response time. Error control codes may be used to correct and detect transmission errors. An interesting issue to investigate is how the choice of error control code affects the mean-variance trade-off.

	Mean Optimal Alg	α-algorithm			Variance Optimal Alg
		2.2	2.6	3	
$\theta = 0.25$	13539.4	13571.1	13653.1	13791.0	14020.9
$\theta = 0.5$	12661.5	12698.1	12816.7	12957.9	13262.5
$\theta = 0.75$	10900.9	10929.1	11104.6	11341.2	11794.2
$\theta = 1.0$	8389.08	8416.24	8685.9	9009.94	9648.67
$\theta = 1.25$	5738.61	5790.42	6092.85	6478.46	7206.52
$\theta = 1.5$	3619.92	3680	3973.95	4372.19	5080.71

Table 5: Mean response time results measured in Experiment 3: Non-uniform Demand, Increasing Lengths

	Mean Optimal Alg	α-algorithm			Variance Optimal Alg
		2.2	2.6	3	
$\theta = 0.25$	9.9772	9.4526	8.7698	8.3946	7.9514
$\theta = 0.5$	10.426	9.7026	8.7773	8.2383	7.7046
$\theta = 0.75$	10.904	9.8431	8.5747	7.8419	7.1803
$\theta = 1.0$	10.450	9.1251	7.6477	6.8327	6.083
$\theta = 1.25$	8.5892	7.2595	5.8089	5.1011	4.4578
$\theta = 1.5$	6.0843	4.8865	3.7022	3.1868	2.8092

Table 6: Variance of response time results measured in Experiment 3: Non-uniform Demand, Increasing Lengths ($\times 10^7$)

	Mean Optimal Alg	α-algorithm			Variance Optimal Alg
		2.2	2.6	3	
$\theta = 0.25$	13397.5	13439.3	13537.6	13651.9	13875.5
$\theta = 0.5$	12516.1	12526.3	12662.5	12844.5	13158.7
$\theta = 0.75$	10701.2	10728.1	10941.7	11172.7	11646.8
$\theta = 1.0$	8138.74	8191.29	8433.63	8750.08	9409.92
$\theta = 1.25$	5464.04	5516.43	5807.97	6172.63	6918.96
$\theta = 1.5$	3355.76	3410.66	3685.77	4049.8	4742.75

Table 7: Mean response time results measured in Experiment 3: Non-uniform Demand, Decreasing Lengths

	Mean Optimal Alg	α-algorithm			Variance Optimal Alg
		2.2	2.6	3	
$\theta = 0.25$	10.006	9.5022	8.8298	8.3807	7.9251
$\theta = 0.5$	10.461	9.7158	8.8018	8.2785	7.6950
$\theta = 0.75$	10.835	9.8293	8.6360	7.8993	7.1786
$\theta = 1.0$	10.303	9.1106	7.6816	6.9136	6.1047
$\theta = 1.25$	8.2702	7.0962	5.8105	5.1400	4.5054
$\theta = 1.5$	5.7515	4.6700	3.6636	3.2175	2.8632

Table 8: Variance of response time results measured in Experiment 3: Non-uniform Demand, Decreasing Lengths ($\times 10^7$)

A Appendix I: Mean and Variance of Response Time

A.1 Derive the expressions for μ

μ is the expected value of random variable t. According to the definition of expected value, we have

$$\mu = \int_0^\infty t g(t) dt \tag{13}$$

¿From Equation (3), we have

$$\begin{aligned} \mu &= \int_0^\infty (t \sum_{i=1}^{M} (p_i q_i(t))) dt \\ &= \int_0^\infty \sum_{i=1}^{M} p_i (t q_i(t)) dt \\ &= \sum_{i=1}^{M} p_i \int_0^\infty t q_i(t) dt \\ &= \sum_{i=1}^{M} p_i \int_0^{s_i} \frac{t}{s_i} dt \\ &= \frac{1}{2} \sum_{i=1}^{M} s_i p_i \end{aligned}$$

A.2 Derive the expressions for σ^2

By the definition in (2), σ^2 is the expected value of random variable $(t-\mu)^2$. So, we have

$$\sigma^2 = \int_{-\infty}^{\infty} (t-\mu)^2 g(t) dt$$

where g(t) is the density function of random variable t.
By substitution, the above equation becomes that

$$\sigma^2 = \int_{-\infty}^{\infty} (t-\mu)^2 \sum_{i=1}^{M} (p_i q_i(t)) dt$$

Then the following derivation is easy to understand.

$$\begin{aligned}
\sigma^2 &= \int_{-\infty}^{\infty} \sum_{i=1}^{M} (p_i (t-\mu)^2 q_i(t)) dt \\
&= \sum_{i=1}^{M} \int_{-\infty}^{\infty} p_i (t-\mu)^2 q_i(t) dt \\
&= \sum_{i=1}^{M} \int_{0}^{s_i} p_i (t-\mu)^2 \frac{1}{s_i} dt \\
&= \sum_{i=1}^{M} \frac{p_i}{s_i} \int_{0}^{s_i} (t-\mu)^2 dt \\
&= \sum_{i=1}^{M} \frac{p_i}{s_i} [\frac{1}{3}(t-\mu)^3 |_0^{s_i}] \\
&= \frac{1}{3} \sum_{i=1}^{M} \frac{p_i}{s_i} [(s_i-\mu)^3 + \mu^3] \\
&= \frac{1}{3} \sum_{i=1}^{M} \frac{p_i}{s_i} (s_i^3 - 3s_i^2\mu + 3s_i\mu^2) \\
&= \frac{1}{3} \sum_{i=1}^{M} (p_i s_i^2 - 3p_i s_i \mu + 3p_i \mu^2) \\
&= \frac{1}{3} \sum_{i=1}^{M} p_i s_i^2 - (\sum_{i=1}^{M} p_i s_i)\mu + (\sum_{i=1}^{M} p_i)\mu^2
\end{aligned}$$

Since $\mu = \frac{1}{2}\sum_{i=1}^{M} s_i p_i$ and $\sum_{i=1}^{M} p_i = 1$, the above equation can be further simplified as

$$\begin{aligned}
\sigma^2 &= \frac{1}{3} \sum_{i=1}^{M} p_i s_i^2 - 2\mu^2 + \mu^2 \\
&= \frac{1}{3} \sum_{i=1}^{M} p_i s_i^2 - \mu^2
\end{aligned}$$

Of course, it can be rewritten in another form without μ, i.e

$$\sigma^2 = \frac{1}{3} \sum_{i=1}^{M} p_i s_i^2 - (\frac{1}{2} \sum_{i=1}^{M} s_i p_i)^2$$

B Appendix II: Minimizing the Variance

Theorem 1 *Given the demand probability p_i of each item i, the minimal* variance of response time, *σ^2, is achieved when the schedule vector possesses the following property, assuming that transmissions of each item i are equally spaced by s_i.*

$$\frac{p_i s_i^2}{l_i}(\frac{2}{3}s_i - \mu) = constant, \forall i, 1 \le i \le M$$

Proof:

From Equation 5, we know that the variance of response time σ^2 is a multi-variable function of a schedule vector's all elements $s_1, s_2, \cdots, s_M$. However, only M-1 of the $s_i^{\prime}s$ can be changed independently instead of M. To find this fact, let us look at the share of bandwidth occupied by each item. For item i, we use the notation r_i to denote the amount of time it takes during the broadcast. Since each transmission of item i takes l_i time and item i is supposed to be transmitted every s_i time period, we have $r_i = \frac{l_i}{s_i}$. As $\sum_{i=1}^{M} r_i = 1$,

$$\frac{l_1}{s_1} + \frac{l_2}{s_2} + \ldots + \frac{l_{M-1}}{s_{M-1}} + \frac{l_M}{s_M} = 1$$

or

$$s_M = l_M(1 - \frac{l_1}{s_1} - \frac{l_2}{s_2} - \ldots - \frac{l_{M-1}}{s_{M-1}})^{-1} \tag{14}$$

Back to our objective of minimizing the σ^2, we have to find the schedule vector which makes $\frac{\partial\sigma^2}{\partial s_i} = 0, \forall i$. We now solve these equations, beginning with $0 = \frac{\partial\sigma^2}{\partial s_1}$.

$$\begin{aligned}
0 &= \frac{\partial\sigma^2}{\partial s_1} \\
&= \frac{\partial}{\partial s_1}(\frac{1}{3}\sum_{i=1}^{M} p_i s_i^2 - \mu^2) \\
&= \frac{\partial}{\partial s_1}[\frac{1}{3}\sum_{i=1}^{M} p_i s_i^2 - (\frac{1}{2}\sum_{i=1}^{M} p_i s_i)^2] \\
&= \frac{\partial}{\partial s_1}[\frac{1}{3}(p_1 s_1^2 + p_2 s_2^2 + \ldots + p_{M-1}s_{M-1}^2 + p_M s_M^2) - \frac{1}{4}(p_1 s_1 + p_2 s_2 + \ldots + p_{M-1}s_{M-1} + p_M s_M)^2] \\
&= \frac{2}{3}p_1 s_1 + \frac{2}{3}p_M s_M \frac{\partial s_M}{\partial s_1} - \frac{1}{4}\cdot 2(p_1 s_1 + p_2 s_2 + \ldots + p_{M-1}s_{M-1} + p_M s_M)(p_1 + p_M \frac{\partial s_M}{\partial s_1})
\end{aligned}$$

$$= p_1[\frac{2}{3}s_1 - \frac{1}{2}(\sum_{i=1}^{M} p_i s_i)] + p_M[\frac{2}{3}s_M - \frac{1}{2}(\sum_{i=1}^{M} p_i s_i)]\frac{\partial s_M}{\partial s_1} \tag{15}$$

From Equation 14, s_M is a function of s_1. So, it can be found that

$$\frac{\partial s_M}{\partial s_1} = -\frac{s_M^2}{s_1^2}\cdot\frac{l_1}{l_M}$$

By substitution, Equation 15 becomes that

$$0 = p_1[\frac{2}{3}s_1 - \frac{1}{2}(\sum_{i=1}^{M} p_i s_i)] - \frac{p_M s_M^2}{s_1^2} \cdot \frac{l_1}{l_M}[\frac{2}{3}s_M - \frac{1}{2}(\sum_{i=1}^{M} p_i s_i)]$$

which implies that

$$\frac{p_1 s_1^2}{l_1}(\frac{2}{3}s_1 - \frac{1}{2}\sum_{i=1}^{M} p_i s_i) = \frac{p_M s_M^2}{l_M}(\frac{2}{3}s_M - \frac{1}{2}\sum_{i=1}^{M} p_i s_i)$$

Similarly,

$$\frac{p_2 s_2^2}{l_2}(\frac{2}{3}s_2 - \frac{1}{2}\sum_{i=1}^{M} p_i s_i) = \frac{p_M s_M^2}{l_M}(\frac{2}{3}s_M - \frac{1}{2}\sum_{i=1}^{M} p_i s_i)$$

$$\cdots$$

$$\frac{p_{M-1} s_{M-1}^2}{l_{M-1}}(\frac{2}{3}s_{M-1} - \frac{1}{2}\sum_{i=1}^{M} p_i s_i) = \frac{p_M s_M^2}{l_M}(\frac{2}{3}s_M - \frac{1}{2}\sum_{i=1}^{M} p_i s_i)$$

In other words,

$$\frac{p_1 s_1^2}{l_1}(\frac{2}{3}s_1 - \frac{1}{2}\sum_{i=1}^{M} p_i s_i) = \frac{p_2 s_2^2}{l_2}(\frac{2}{3}s_2 - \frac{1}{2}\sum_{i=1}^{M} p_i s_i) = \ldots = \frac{p_M s_M^2}{l_M}(\frac{2}{3}s_M - \frac{1}{2}\sum_{i=1}^{M} p_i s_i)$$

This is equivalent to saying that

$$\frac{p_i s_i^2}{l_i}(\frac{2}{3}s_i - \frac{1}{2}\sum_{i=1}^{M} p_i s_i) = constant, \forall i, 1 \le i \le M$$

or

$$\frac{p_i s_i^2}{l_i}(\frac{2}{3}s_i - \mu) = constant, \forall i, 1 \le i \le M$$

Thus, we have proved Theorem 1.

C Appendix III: Lower Bounds for the α-Algorithm

In the ideal situation, α-algorithm can create a schedule making the equation $\frac{s_i^\alpha p_i}{l_i} = C$ to be true, where C is a constant. In the following, we will derive the value of C, and the values of s_i's when the ideal condition holds. Then, both the mean and variance of response time can be obtained. They serve to be the lower bounds of mean and variance of response time respectively, which can be attained by an α-algorithm.

From the equation $\frac{s_i^\alpha p_i}{l_i} = C, i = 1, 2, \cdots, M$, it follows that

$$s_i = (C \cdot \frac{l_i}{p_i})^{\frac{1}{\alpha}} \tag{16}$$

Let r_i be the share of bandwidth by item i during broadcast. Since each transmission of item i takes l_i time and item i is transmitted every s_i time period, we have $r_i = \frac{l_i}{s_i}$. As $\sum_{i=1}^{M} r_i = 1$,

$$\sum_{i=1}^{M} \frac{l_i}{s_i} = 1 \tag{17}$$

Substituting the s_i in above equation with the expression in Equation 16, we have

$$\sum_{i=1}^{M} \frac{l_i}{(C \cdot \frac{l_i}{p_i})^{\frac{1}{\alpha}}} = 1$$

or

$$\frac{\sum_{i=1}^{M}(p_i^{\frac{1}{\alpha}} l_i^{1-\frac{1}{\alpha}})}{C^{\frac{1}{\alpha}}} = 1 \tag{18}$$

Solving the equation above, we get

$$C = \left(\sum_{i=1}^{M}(p_i^{\frac{1}{\alpha}} l_i^{1-\frac{1}{\alpha}})\right)^{\alpha} \tag{19}$$

Substituting the value of C into Equation 16, we find the value of s_i for item i as follows,

$$s_i = \sum_{i=1}^{M}(p_i^{\frac{1}{\alpha}} l_i^{1-\frac{1}{\alpha}}) \; (\frac{l_i}{p_i})^{\frac{1}{\alpha}}, i = 1, 2, \cdots, M$$

Finally, the values of mean μ and variance σ^2 in this case can be derived by substituting the above expression for s_i into Equations 4 and 5.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks - data management for asymmetric communications environment," in *ACM SIGMOD Conference*, May 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik, "Balancing push and pull for data broadcast," in *ACM SIGMOD Conference*, May 1997.

[3] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-based data delivery using broadcast disks," *IEEE Personal Communication*, pp. 50–60, Dec. 1995.

[4] M. H. Ammar, "Response time in a teletext system: An individual user's perspective," *IEEE Transactions on Communications*, Nov. 1987.

[5] S. Hameed and N. H. Vaidya, "Log-time algorithms for scheduling single and multiple channel data broadcast," in *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Sept. 1997.

[6] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Data on the air - organization and access," *IEEE Transactions of Data and Knowledge Engineering*, July 1996.

[7] J. Milton and J. C. Arnold, *Introduction to Probability and Statistics.* McGraw-Hill, Inc., 1995.

[8] C.-J. Su and L. Tassiulas, "Broadcast scheduling for the distribution of information items with unequal length," *Technical Report*, 1997.

[9] N. H. Vaidya and S. Hameed, "Data broadcast in asymmetric wireless environments," in *Workshop on Satellite Based Information Services (WOSBIS), Rye, NY*, Nov. 1996.

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