TECHNICAL RESEARCH REPORT

Broadcast Scheduling for Push Broadcast Systems with Arbitrary Cost Functions

by Majid Raissi-Dehkordi, John S. Baras

TR 2007-5

ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the Glenn L. Martin Institute of Technology/A. James Clark School of Engineering. It is a National Science Foundation Engineering Research Center.

Web site http://www.isr.umd.edu

Broadcast scheduling for Push broadcast systems with arbitrary cost functions

Majid Raissi-Dehkordi* and John S. Baras
Institute for Systems Research and Department of Electrical and Computer Engineering
University of Maryland, College Park, Maryland 20742
Email: majid@isr.umd.edu, baras@isr.umd.edu

Abstract

In this report, the problem of broadcast scheduling in *Push* broadcast systems is studied. We introduce an optimization approach that leads to well justified policies for Push broadcast systems with generalized cost functions. In particular, we apply our results to a Push broadcast system with different deadlines associated to the files while allowing the files to have unequal demand rates and lengths. We will show that our proposed policy covers some of the previously investigated Push systems as special cases and is applicable to a wide range of cost functions assigned to the files. We also calculate the optimal average cost for our experimental settings and show, through extensive simulation studies, that our results closely match that value for each experiment.

Index Terms

Broadcast Information Delivery, Push Broadcast,Bulk Queueing Systems.

I. Introduction

The increasing demand for various information delivery services in recent years, together with advances in wireless network technology, have resulted into numerous research works on more efficient methods for the delivery of information. In most information delivery applications the flow of data is not symmetric. In a typical data delivery application, there are a few information sources and a large number of users, thus, the volume of data transferred from the server to the users (*downlink*) is much larger than that in the reverse direction (*uplink*) which can be zero as we will explain. News, weather, traffic and stock quotes are examples of the types of information that can be provided by these applications as opposed to applications with one-to-one information content such as *email*. Although the above services can be, and in fact are, already implemented over terrestrial links, it is their combination with wireless technologies that unfolds the real potential of these applications. The mobility of the wireless devices and the capability for accessing the information any time and anywhere is one of the main forces behind the demand for such services. Another aspect of wireless data delivery systems which is the primary focus of this paper is the inherent broadcast nature of wireless communications. If the data to be transmitted is of high demand it is possible to send every data package to all the users who need it at the same time and over a single broadcast channel. The efficiency of this method and the amount of downlink bandwidth that is saved depends on the number of users awaiting that information at the same time or, in other words, the popularity of the information.

Research partially supported by NASA cooperative agreement NCC3-528

This fact translates into a high degree of scalability for such systems. Given that the geographical coverage and uplink bandwidth (if needed) are adequately provided, the downlink bandwidth is independent of the number of users in the system which makes such systems highly scalable and economical.

The two main architectures for broadcast delivery are the one-way (*Push*) and the on-demand (*Pull*) systems. The two systems differ in the lack or presence of a return channel to transfer the user requests to the server. In a Push system, the server does not actually receive the requests and schedules its transmissions based on the statistics of the user request pattern (hence the term *Push*). Conversely, in a *Pull* system, the server receives all the requests and can schedule the transmissions based on the number of requests for different data packages. A Pull system is potentially capable of achieving a higher performance than a Push system but the cost of a return channel can generally overshadow this performance improvement. For this reason *hybrid* architectures, those that combine Push and Pull systems, are commonly suggested in the literature [1]–[3]. The main problem with both of the above broadcast methods is the scheduling of data file transmissions to minimize a cost function. Based on the nature of the applications supported by a data delivery system, different cost functions can be introduced to model the cost of delayed delivery of the information to the users.

This report addresses the scheduling problem in a Push broadcast system. It aims to find the *optimal* (with respect to the generalized average cost) scheduling policy and also provides a benchmark for evaluating current and possibly future heuristic algorithms. Our main contribution is deriving a solution that allows arbitrary cost functions to be assigned to the information files. Specifically, we work on the systems where different *deadlines* are assigned to different files and introduce policies that minimize the average tardiness over all users.

The main body of previous research on this subject has been concentrated on policies that minimize the average waiting time over all users. However, at least for certain types of information, pure delay can be too simplistic a measure for cost representation. For example, for the users of the information about stock quotes, only a small amount of delay can be acceptable and the information starts to lose its value after that delay. On the other hand, for the users of weather information, a larger delay is acceptable and the information keeps its value for a longer time. This and other similar facts are the main rationale for this research i.e., to address the scheduling problem in a general setting. We approach the scheduling problem from an optimization point of view and derive a lower bound on the achievable average cost by relaxing some of the constraints of the problem. We then use the results and the form of the optimal policy for that problem to derive scheduling policies for our original problem and verify the effectiveness of our solution by simulation studies and comparing the results with the lower bound cost.

This paper is organized as follows. In Section II the exact formulation of the problem is presented and the previous works on this subject are reviewed. In Section III our optimization approach to the problem and the proposed scheduling policy are explained. Section IV is dedicated to performance evaluation of our policy.

II. Push broadcast scheduling, formulation and previous work

In a typical Push broadcast system N separate information files are stored in the system. The request arrival process for each file is modeled by a Poisson and we denote by λ_i the rate of the process for file $i;\ i = 1, \ldots, N$. Each arrival process defines the times when the users decide to access the corresponding file. We also assume that every file $i;\ i = 1, \ldots, N$ has a length l_i which indicates the time to transmit the file over a channel with unit bandwidth. In a Push system, the only information available to the scheduler about the requests is the arrival rates $\lambda_i;\ i = 1, \ldots, N$ known beforehand and it does not have any knowledge of the request generation times or the

number of pending requests for the files at any time. The term *request* in this paper is used to specify the times when the users need the files even though no real request is generated. We define a cost function $C_i(t);\ i = 1, \ldots, N$ associated with each file. The cost function is assigned to every user who requests file i and shows the amount of cost incurred by the system for that user if there is a time delay t between the request time and the broadcast time of the file. The system is non-preemptive, meaning that files are transmitted as a whole and the broadcast of a file can not be interrupted. Therefore, the broadcast of file i will take exactly l_i seconds.

The scheduling problem is defined as coming up with a schedule for broadcasting the N files over time such that the total average cost incurred by the system is minimized. In our setting, the users who need a file after the beginning of its current transfer , need to wait for the next complete broadcast to receive the file. Also, the waiting times in our formulation are defined as the time between the user request time and the *start* of the broadcast of the file. A slightly different setting is to define the waiting time till the *end* of the broadcast which results into policies more favorable to files with longer lengths. We address the problem using the first definition of the waiting time but the same approach can be readily applied to the other definition as well.

If we denote by $\bar{C}_i$ the long-term average cost incurred from the users requesting page i when a particular scheduling policy is in use, the overall average cost can be written as

$$C = \frac{1}{\lambda} \sum_{i=1}^{N} \lambda_i \bar{C}_i \tag{1}$$

where $\lambda = \sum_{i=1}^{N} \lambda_i$.

Although the problem of broadcast scheduling has not been considered in the above general form before, there has been a number of rather interesting works on both the Push and Pull systems with average waiting time objective function i.e. $C_i(t) = t,\ i = 1, \ldots, N$. One of the earliest works on the Push broadcast scheduling is the work by Ammar and Wong [4], [5] where they studied a system with all files having a unit length and showed that the optimal scheduling policy has a periodic form. In other words, the minimum average delay is achieved when the broadcast instances of each file are equally spaced on the channel with a proper period specific to each file. They also showed that the minimum average delay is achieved if the inter-broadcast periods of any two files i and j on the broadcast channel are related as

$$\frac{\tau_i}{\tau_j} = \sqrt{\frac{\lambda_j}{\lambda_i}} \tag{2}$$

i.e., the files with lower request arrival rates are broadcasted less often (larger periods). Finding a schedule that maintains the periodicity of broadcasts for all files and, at the same time, meets the requirement of equation (2) is not always possible. Therefore, they also introduced a heuristic method for designing one period of the broadcast cycle. Given a length L for the cycle, they try to fit proper number of appearances for all file in that cycle to satisfy the optimality equation as closely as possible. The final broadcast schedule is then achieved by repeating that cycle over and over. In a later work [6], it was shown via optimization arguments that if the files have different sizes $l_i;\ i = 1, \ldots, N$, equation (2) is extended as

$$\frac{\tau_i}{\tau_j} = \sqrt{\frac{\lambda_j}{\lambda_i}} \sqrt{\frac{l_i}{l_j}}. \tag{3}$$

Instead of an off-line procedure for cycle design, a real-time heuristic policy for achieving near-optimal results was introduced. A real-time policy is defined as a policy that chooses a file for broadcast based on the information

available at the end of the current broadcast. In another work, Su and Tassiulass [7] considered the system with equal length files and proposed a real-time policy for broadcasting the files over a broadcast channel. In their work, they proposed a parametric real-time scheduling policy and optimized the value of the parameter through a number of simulation experiments. This policy can be regarded as a special case of the policy given in [6] when all file lengths are set to one. The same experimental approach has been used in [8] to find a real-time policy for unequal file lengths that basically results in the same policy derived in [6]. The interested reader is also referred to other publications on this subject such as [9]–[14] and references therein for a more diverse review of the problem.

III. Our approach

All of the previous works on Push broadcast systems have studied systems with the average waiting time criteria where the cost function for each user is the elapsed time from the time the user needs a specific file till the next broadcast of that file. In general, the cost function is not necessarily the $c(t) = t$ line and depending on the user behavior and nature of the requested file it may take different shapes. In this section, we will address the problem of broadcast scheduling in Push systems in its general form when the cost function is any monotonic non-decreasing function of time and present an optimization approach to find near-optimal scheduling policies. We will then apply our method specifically to a system where the files have different lengths and the cost function represents the waiting time as well as the *deadline* associated with each file. This case can be also looked at in a different context by considering a system where the user's device constantly receives and stores the broadcasted files of interest and the user always accesses the most recent version stored in the device. The delay in that case is in fact the *age* of the file stored in user's system and the cost is a function of that age. For cases such as traffic information or stock quotes, there is usually a certain amount of delay that is acceptable and the cost starts to increase only after that period has passed. It is not difficult to show that the scheduling problems for that type of system will reduce to the same problem as our original system. Either way, to our knowledge, the broadcast scheduling problem in Push systems has not been addressed in its generality and there are no policies that address the problem beyond the average waiting time criteria particularly, for the rather important case when deadlines are involved.

Our approach for addressing this specific scheduling problem is as follows. First, we consider the scheduling problem in a system similar to our system but with weaker constraints. We call the new system the *relaxed* system and the scheduling problem associated with it the relaxed problem since they are obtained by relaxing some of the constraints of the original system. After finding the optimal solution for the new problem, we use that to come up with a scheduling policy for the original system. Our assumptions and notations are as follow:

- N: Total number of files stored in the system
- The demand generation process for each file i is a Poisson process with known rate $\lambda_i;\ i = 1, \ldots, N$
- l_i: Length of file i
- $C_i(t)$: Cost function associated with each of the demands for file i as a function of the delay between demand generation time and the next broadcast of the file. It is always assumed that for all i, $C_i(t) = 0$ if $t \leq 0$.
- It is assumed without loss of generality that the total channel bandwidth is 1 and only one file can be in broadcast at any time (Time Division Multiplexing)

In the relaxed problem, we assume that the instantaneous bandwidth is not limited to 1 and only the long-term average of the total used bandwidth should not exceed this value. This assumption is similar to a relaxation made in [15] and [16] in a Dynamic Programming approach to the broadcast scheduling problem in Pull systems and

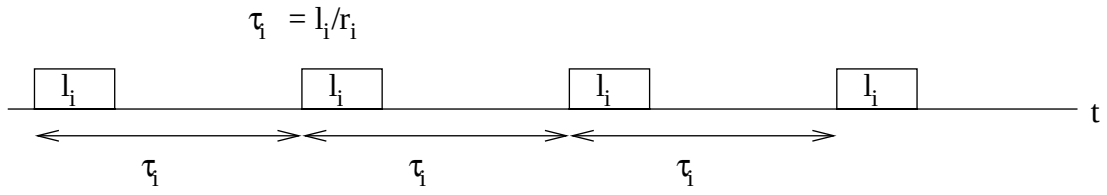

Fig. 1. optimal broadcast schedule on a single channel

introduced in [17] in the general context of Restless Bandit Problems. Another way to look at the new problem is to assume a system with N parallel unit-bandwidth channels, one for each file, with the only restriction that in the long run, the total used bandwidth should be less than or equal to 1. It is not difficult to show that the optimal solution should always fully utilize the bandwidth i.e., meeting the constraint with equal sign. Since our original problem with the strict constraint on the instantaneous bandwidth is a special case of the new problem, the optimal cost for this new system is obviously a lower bound for the original system. This will later allow us to compare the performance of our policies with this lower bound and find out how well they perform.

Let's denote by r_i the average long-term bandwidth assigned for broadcast of file i. The only constraint is then to have

$$\sum_{i=1}^{N} r_i \leq 1. \tag{4}$$

Having r_i values fixed for each file $i,\ i = 1, \ldots, N$, it can be shown that the average cost is minimized with a periodic broadcast schedule for each channel.

Theorem 1: For a single file with length l, average broadcast bandwidth r and monotonic non-decreasing cost function $C(t)$ over $t \geq 0$, the minimum average cost is achieved when the successive broadcast instances of the file follow a periodic pattern.

Proof: See appendix I.

This result, in the special case of average waiting time criteria (i.e., $C(t) = t$), is in line with the well-known results (e.g. [4], [5]) about the optimality of periodic schedules.

Given the length l_i and allocated bandwidth r_i for a typical file i, the broadcasts happen with a period $\tau_i = l_i/r_i$ (figure 1). For such a periodic schedule, the long-term average cost for each file is equal to the average cost per period given by

$$\bar{C_i} = \frac{1}{\tau_i} \sum_{n=0}^{\infty} \frac{e^{-\lambda_i \tau_i} (\lambda_i \tau_i)^n}{n!} n c_i = \lambda_i c_i \tag{5}$$

where c_i is the average cost incurred by each user which is

$$c_i(\tau_i) = \frac{1}{\tau_i} \int_0^{\tau_i} C_i(t) dt.$$

With the above definitions, the scheduling problem can be formulated as a constrained optimization problem as follows

$$\min_{\tau_1, \ldots, \tau_N} \frac{1}{\lambda} \sum_{i=1}^{N} \lambda_i c_i$$

$$\text{s.t.} \quad \begin{cases} \sum_{i=1}^{N} \frac{l_i}{\tau_i} \leq 1 \\ \tau_i > 0, \qquad \forall i \in \{1, \ldots, N\} \end{cases}$$

where $\lambda = \sum_{i=1}^{N} \lambda_i$. From a practical point of view, since we have not assigned any cost for using the channel, it is obvious that the optimal policy would make full use of the channel bandwidth. Equivalently, it means that the optimal solution of the above problem occurs on the border of the constraint space i.e., when $\sum \frac{l_i}{\tau_i} = 1$. The exact statement of this fact is as follows

Theorem 2: If all $C_i(.)$s are monotonic non-decreasing, then the solution $(\tau_1^*, \ldots, \tau_N^*)$ for the above optimization problem occurs on the boundary of the constraint space i.e., we have $\sum \frac{l_i}{\tau_i^*} = 1$.

Proof: Appendix II.

We can now use the Lagrange method to find the optimal solution. Let's introduce the *relative demand* parameters $q_i = \lambda_i/\lambda$ for $i = 1, \ldots, N$. we need to find

$$\min_{\tau_1,\ldots,\tau_N,\mu} L \tag{6}$$

where

$$L(\tau_1, \ldots, \tau_N, \mu) = \sum_{i=1}^{N} q_i c_i + \mu \left(\sum_{i=1}^{N} \frac{l_i}{\tau_i} - 1 \right). \tag{7}$$

the optimal solution satisfies

$$q_i \frac{dc_i}{d\tau_i} - \frac{\mu l_i}{\tau_i^2} = 0; \; i = 0, \ldots, N \tag{8}$$

$$\sum_{i=1}^{N} \frac{l_i}{\tau_i} = 1. \tag{9}$$

The above $N + 1$ equations can be solved to find the optimal values of τ_1 to τ_N resulting in the minimum total average cost for arbitrary cost functions. These equations place a requirement on the period τ_i that depends on the length and demand rate of file i

$$\frac{q_i \tau_i^2}{l_i} \frac{dc_i}{d\tau_i} = \mu. \tag{10}$$

or

$$\frac{\tau_i^2}{\tau_j^2} = \frac{q_j}{q_i} \frac{l_i}{l_j} \frac{dc_j/d\tau_j}{dc_i/d\tau_i}. \tag{11}$$

The above constraints are our guidelines for coming up with a solution for the original system where only one file at any time can be in broadcast over the single available channel. One approach is to design a single period of the overall periodic broadcast schedule and arrange multiple broadcast instances of each file inside one period to satisfy equation (11) as closely as possible. The long-term schedule would then consist of successive repetitions of this single period. This approach has been used in [5] and [6]. A second approach is to treat the schedule design as a dynamic scheduling problem despite the fact that the system is completely deterministic as long as the demand rates are constant. We use this approach throughout this paper since it removes the problem of designing a full cycle at the expense of some decision overhead at the end of each broadcast. If the complexity of the decision policy can be kept low, this approach can be easily implemented in real systems.

Let's define t_i to be the time elapsed since the last broadcast of file i. In the ideal case, file i is broadcasted each time t_i hits the τ_i value which satisfies equation (10) as shown in figure (2). However, since for some files, t_i may reach the optimal value while another file is in transmission, several files may have their t_i values passed the optimal value at the end of the current broadcast. We therefore use the $\frac{q_i t_i^2}{l_i} \frac{dc_i}{d\tau_i} - \mu$ value for each file as the *eligibility* of that file for broadcast at the current decision time. Since any monotonic increasing function of our

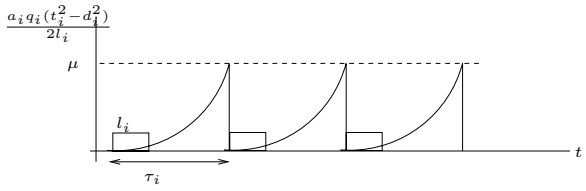

Fig. 2. optimal broadcast schedule on a single channel

eligibility measure can also be used as the index function, we can use the index policy that assigns the index function

$$\nu_i = \frac{q_i t_i^2}{l_i} \frac{dc_i}{d\tau_i} \tag{12}$$

to each file and selects the file with the largest ν_i value for broadcast. The only information required by the policy about the state of the system are the last broadcast time for each of the pages since the t_i values would simply be the current time minus those times. The complexity of this policy is $O(N)$ since index functions are calculated independently and the largest among them is chosen.

Having established a general framework for index policies for the Push broadcast systems, we can now be more specific and consider systems with specific cost functions.

A. Systems with average waiting time criteria

The cost function $C_i(.)$ for this type of systems is defined as

$$C_i(t) = a_i t,\ i = 1, \ldots, N$$

where a_i is a weighting factor for page i. This case covers almost all of the Push broadcast systems analyzed in previous research on this subject. For this case we have

$$\begin{aligned} c_i &= \frac{1}{\tau_i} \int_0^{\tau_i} a_i t dt \\ &= \frac{1}{2} a_i \tau_i. \end{aligned}$$

Therefore the optimal policy for the relaxed system satisfies (equation 10)

$$\frac{a_i q_i \tau_i^2}{2l_i} = \mu \tag{13}$$

or

$$\frac{\tau_i}{\tau_j} = \frac{\sqrt{q_j}}{\sqrt{q_i}} \frac{\sqrt{l_i}}{\sqrt{l_j}} \frac{\sqrt{a_j}}{\sqrt{a_i}},\ i, j \in \{1, \ldots, N\}$$

which reduces to equation (3) [6] when we further set $a_i = 1$ for $i = 1, \ldots, N$. The index function in this case is

$$\nu_i = \frac{q_i a_i t_i^2}{l_i}$$

which reduces to the on-line policy introduced in [6] for the special case where $a_i = 1$ and to the policy in [7] when all files have unit length i.e., $l_i = 1$ for all i.

By plugging in the optimal values of τ_i from equation (13) in equation (9), the optimal value of μ and subsequently the total average waiting time C are found to be

$$\mu = \frac{1}{2} \left(\sqrt{a_i l_i q_i} \right)^2 \tag{14}$$

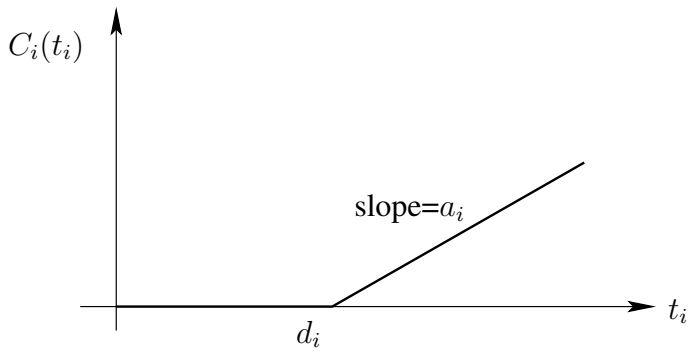

Fig. 3. Cost function for the tardiness criteria

and

$$C = \frac{1}{2}\left(\sum_{i=1}^{N}\sqrt{a_i l_i q_i}\right)^2. \tag{15}$$

This value serves as a lower bound for the optimal value achievable by policies for the original system and we will use it as a reference in our performance evaluation studies.

B. Systems with average tardiness criteria

When the timeliness of receiving the information is of importance and each file has its own expiration period, the average waiting time is no longer a good criteria. A criteria that is often used for such cases (e.g. [18]) is the average tardiness criteria. The cost function representing the tardiness is shown in figure 3 where the cost starts to grow with slope a_i only after a deadline associated with the file is passed. With this definition, the average delay criteria is in fact a special case of the average tardiness criteria when all $d_i = 0$. For the tardiness cost function we have

$$c_i = \frac{1}{\tau_i}\int_0^{\tau_i} C_i(t)dt$$

or

$$c_i = \begin{cases} \frac{1}{2}a_i\frac{(\tau_i - d_i)^2}{\tau_i} & \tau_i \geq d_i \\ 0 & \tau_i < d_i. \end{cases} \tag{16}$$

which implies (assuming $\tau_i > d_i$)

$$\frac{dc_i}{d\tau_i} = \frac{a_i}{2}\frac{\tau_i^2 - d_i^2}{\tau_i^2}. \tag{17}$$

Using equation 10 we find

$$\frac{a_i q_i\left(\tau_i^2 - d_i^2\right)}{2l_i} = \mu \tag{18}$$

or

$$\tau_i = \sqrt{\frac{2l_i\mu}{a_i q_i} + d_i^2} \tag{19}$$

and the index function can be defined as

$$\nu_i = \frac{a_i q_i\left(t_i^2 - d_i^2\right)}{l_i}. \tag{20}$$

According to this function, in otherwise similar conditions, a more popular file (larger q_i) is given priority over a lesser popular file. Similarly, a more time-critical file (small d_i) has priority over another file with a larger deadline.

The length however, has a negative impact on the priority and a file with a shorter length is chosen over a longer file in similar conditions. Also note that this index policy does not contain τ_i or μ terms and therefore there is no need for explicit calculation of those quantities by the system.

Here we have implicitly assumed that the average total bandwidth is smaller than that needed for periodic broadcast of all files right on their deadline expiration times i.e., file i being broadcasted every d_i seconds. Otherwise, the problem would have been trivial. This assumption can be expressed as

$$\sum_{i=1}^{N} \frac{l_i}{d_i} > 1. \tag{21}$$

Combining equations (9) and (19) results in the following equation for μ

$$\sum_{i=1}^{N} \frac{1}{\sqrt{\frac{2\mu}{a_i l_i q_i} + \left(\frac{d_i}{l_i}\right)^2}} = 1. \tag{22}$$

This equation can be solved to find the optimal value of μ. A look at this equation for μ shows that for $\mu = 0$ the left hand side of the equation is greater than the right hand side due to equation (21). Also, it is obvious that the left side approaches zero as μ approaches infinity. Since the derivative of the left side with respect to μ is always negative, then the solution for μ is unique and greater than zero. This guarantees that, based on equation (18), every τ_i is greater than its corresponding d_i and therefore the assumption of c_i taking the first case in equation (16) is valid. It is also intuitively clear from figure 3, and not difficult to show that the optimal solution should be sought in the $\tau_i \geq d_i$ region since nothing is gained by going to the $\tau_i < d_i$ region for any i.

The optimal τ_i values can now be computed from μ using equation (18) and the optimal value of the total average tardiness would be

$$C = \frac{1}{2} \sum_{i=1}^{N} q_i a_i \frac{(\tau_i - d_i)^2}{\tau_i}. \tag{23}$$

Again, this value is a lower bound for the original system where the bandwidth can not exceed 1, even instantaneously.

It is constructive, as a side note at this point, to briefly discuss those systems where the delay for every user is defined from the request time till the *end* of the transmission of the requested file. This new delay is equal to the delay defined for our original system plus the length of the requested file. This means , for the average tardiness case, that the length l_i is effectively subtracted from the deadline d_i to result into a new smaller deadline d_i'. The problem is then similar to our original problem though with slightly different assumptions for the existence of an optimal solution.

C. Multi-channel Broadcast systems

Our approach can be also easily applied to systems with more than one broadcast channels. Let's assume that the system has K parallel channels $(K < N)$ each with bandwidth 1. We also assume that all users are able to listen to all K channels and receive information from them. However, we rule out the possibility of receiving different parts of a file from more than one channel. The scheduling problem for the relaxed system in this case can be

written as

$$\min_{\tau_1,\ldots,\tau_N} \frac{1}{\lambda}\sum_{i=1}^{N}\lambda_i c_i$$
$$s.t.$$
$$\sum_{i=1}^{N}\frac{l_i}{\tau_i} \leq K$$

with the assumption that

$$\sum_{i=1}^{N}\frac{l_i}{d_i} > K. \tag{24}$$

Following the same approach, the optimality equations for the τ_i values will still be the same as equation (18) but with a different value for μ which is , for the average tardiness criteria, the solution of the following equation

$$\sum_{i=1}^{N}\frac{1}{\sqrt{\frac{2\mu}{a_i l_i q_i}+\left(\frac{d_i}{l_i}\right)^2}} = K. \tag{25}$$

Based on our previous discussions about the properties of this equation, the new solution for μ is smaller than the solution for the single-channel case. This effectively means that although all τ_i values maintain their relationship with each other, they will all take larger values than before, resulting in a smaller average cost for each file, and also a smaller total average cost. The index policy for the original problem is therefore to calculate the same index function for each file as before and broadcast the files with the first K largest index values.

Having derived our scheduling policy, we can now discuss its performance through a number of simulation studies.

IV. Performance Results

In order to evaluate the performance of our policy, we set up a Push broadcast system with 100 files. We set the total demand rate λ as a variable and pick q_i values according to a Zipf distribution [19] with unit exponent i.e., $\forall i,j:\ q_i/q_j = j/i$ and $\sum q_i = 1$.

Since the policy allows the files to have different parameter values i.e., different q_i, d_i, l_i and, a_i values, from each other, our experiments are broken into several sets. In each set, one or more parameters are set equal across all files to allow for a clear evaluation of the policy. Also, here we only focus on cases with $d_i > 0$, at least for some files, since otherwise we would be dealing with a system with average delay criteria and our policy was found to be a reconfirmation of the policy in [6] and only its extension for allowing unequal weights assigned to the files. The interested reader is referred to the performance evaluation studies in that reference for that special case.

As an initial study, we eliminate the effect of file lengths and file weights by setting all l_i and a_i values to one. We also set all deadlines to a common value d and run the experiments by changing the total request arrival rate λ from 15 to 50 with a step of 5 and setting the common deadline value to $d = 1$, $d = 2$ and $d = 3$. Figure 4 shows the total average tardiness obtained from the experiments along with the lower bound values calculated for each experiment. Our first observation is that the average cost is independent of the total request arrival rate as long as the ratios between the arrival rates are kept unchanged. This is an expected result due the fact that the contribution of the arrival rates λ_i to our policy and the expected cost values is only through the normalized values q_i and independent of their true values. Based on this observation, the total arrival rate in all of our following experiments

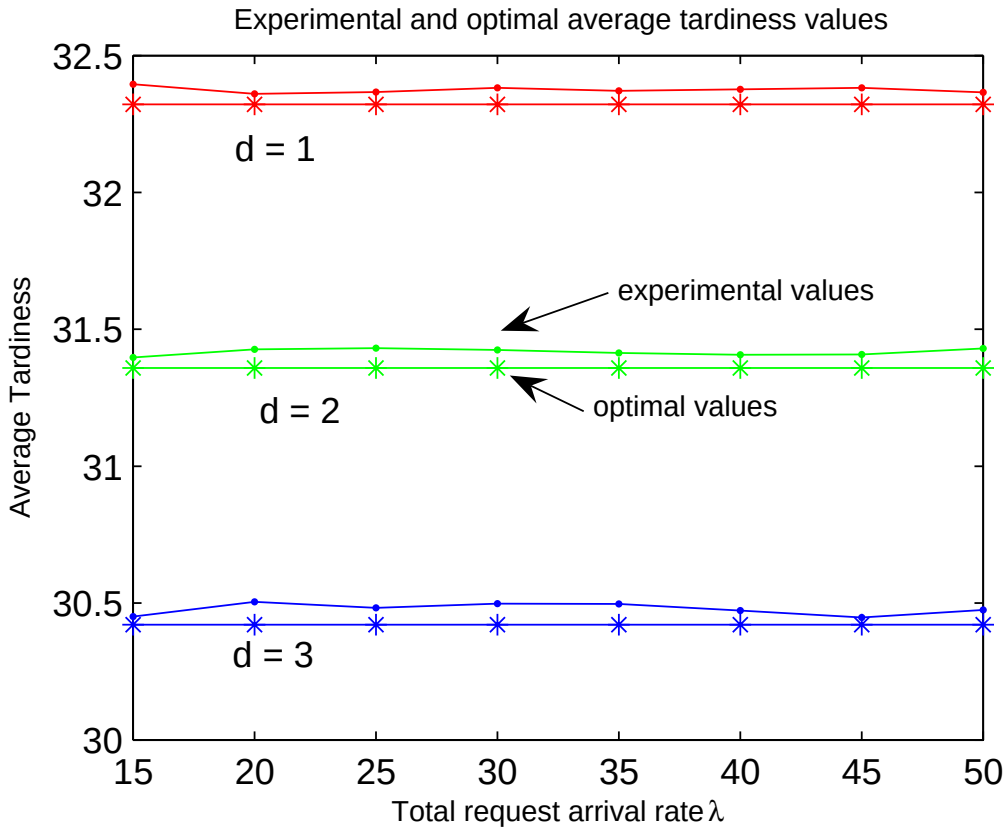

Fig. 4. Optimal and experimental average tardiness for a system with equal deadlines assigned to all files

is set to a fixed value of 50 and is not changed. The second observation in this experiment is the small difference between the experimental results and the corresponding optimal values. As we expect, for a fixed total bandwidth, shorter deadlines result in larger average costs.

Since the average cost values are different for different experiments, we need to define a unified method to evaluate the results of all experiments. Since we are able to calculate the optimal value of the average tardiness for the relaxed problem and it is a lower bound for our real problem, the *goodness* measure in our experiments is defined as how close we get to that value. If we denote by $\hat{C}$ the average tardiness resulting from our heuristic policy and by C the lower bound average tardiness given by equation (23), the *goodness* measure G is defined as

$$G = 100 * \frac{\hat{C} - C}{C}. \tag{26}$$

Figure 5 shows the values of G for the above experiments. It is clear from the graph that the results obtained from our policy are less than 0.3% different from the optimal values for those experiments.

In the second set of experiments, both the deadlines and file lengths are varied and the performance of the policy under different assignment methods for both quantities is evaluated. The total arrival rate is fixed at 50 and all weights are set to one. The deadlines are assigned to the files in three different ways. In the first set of experiments, all deadlines are equal to a fixed number d . In the second set, the deadline assignment is a linear function of the file number with a positive slope such that $d_i = (i-1)/10 + d$ for $i = 1, \ldots, 100$. In the third set of experiments, the deadlines are assigned in the reverse order such that $d_i = (99-i)/10 + d$. The same set of options is applied to the file lengths as well such that file lengths take a constant value l, as well as increasing and decreasing values

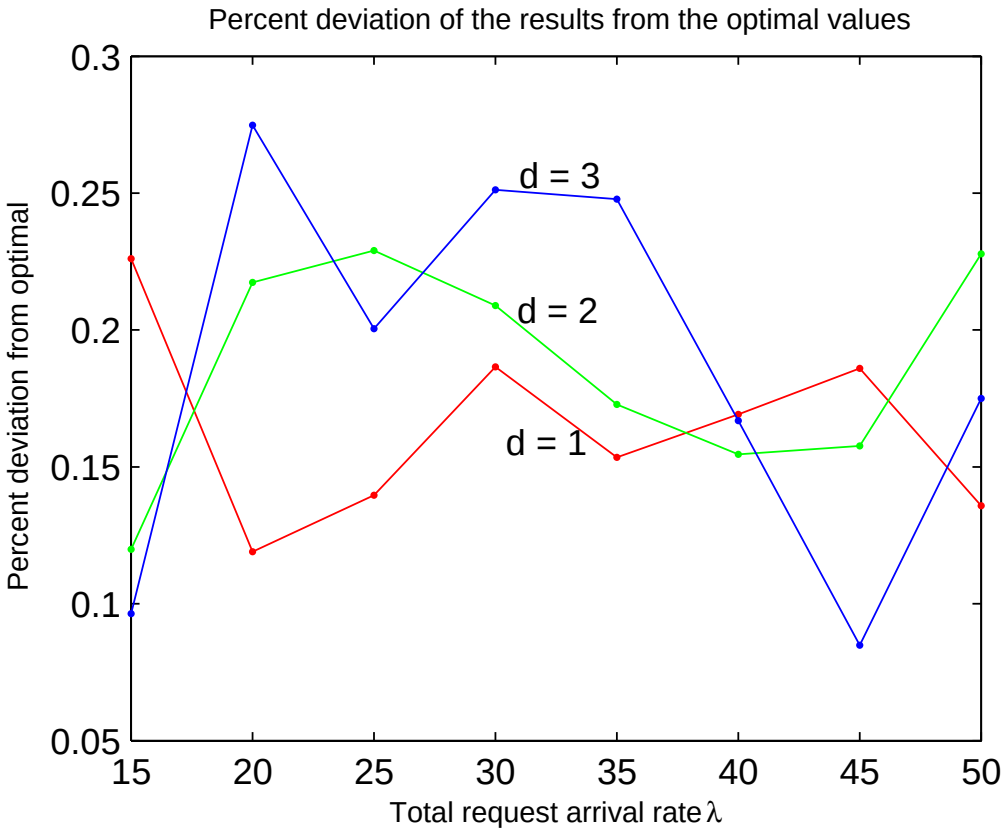

Fig. 5. Relative deviation of the results from the optimal values

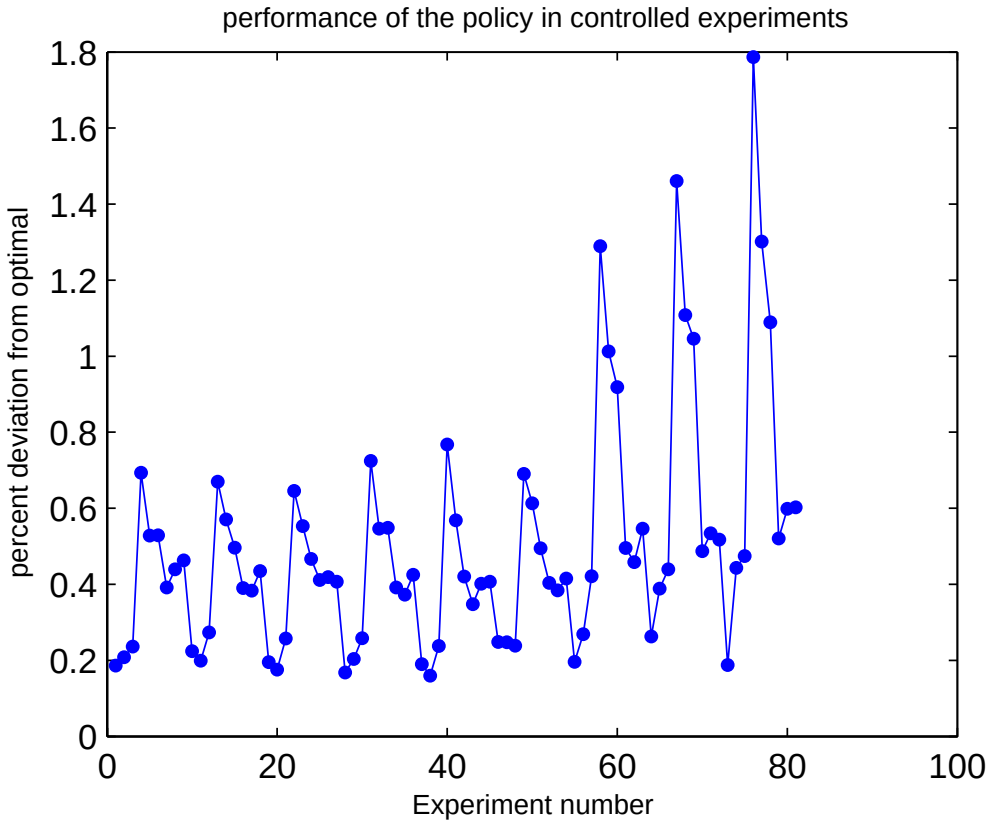

Fig. 6. Relative deviation of the results from the optimal values for controlled assignment of deadlines and lengths

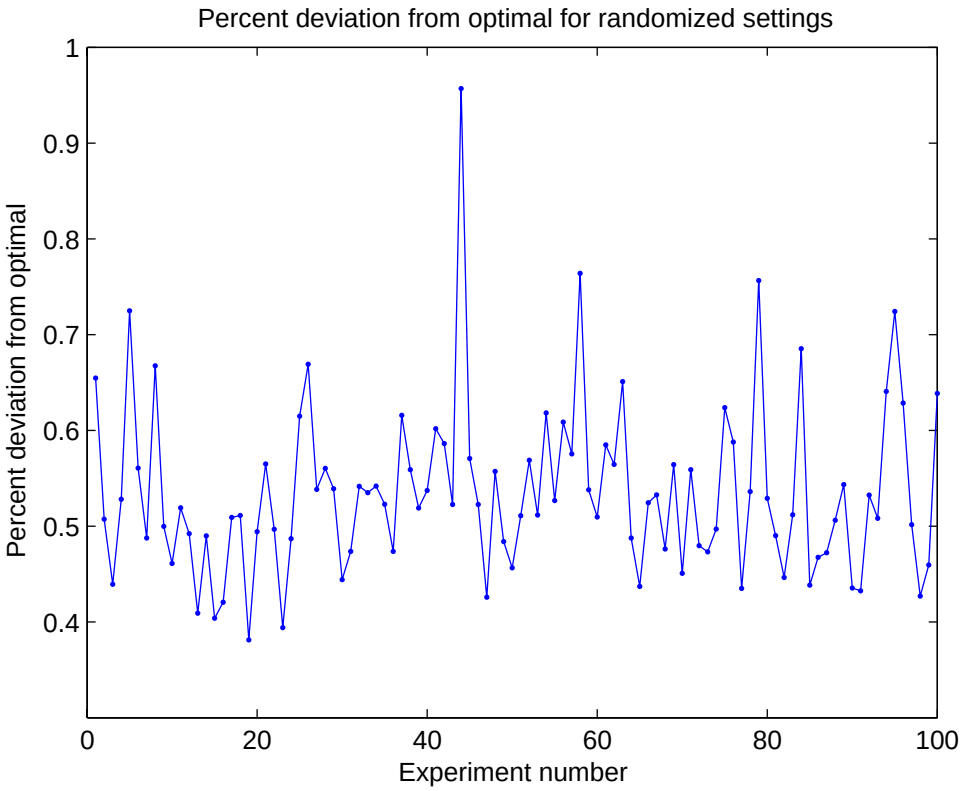

Fig. 7. Relative deviation of the results from the optimal values for randomized assignment of parameters

with offset l. The goal is to find out how the policy performs in the above cases and discover worst case conditions for its performance. For these experiments, both d and l take values from $\{1, 2, 3\}$ resulting in a total number of 81 experiments. Figure 6 shows the G values for those experiments. The worst case scenarios can be easily located as the nine points on the right half of the graph. Those points in fact correspond to experiments in which the deadlines are assigned in the decreasing order of file numbers while the file lengths are assigned in the increasing order. In other words, the most popular file has the longest deadline and shortest length while the least popular file has the shortest deadline and the longest length. The three peak points among these nine experiments are those where the file length offset is minimum ($l = 1$) and the highest peak is when the deadline offset is also at its maximum value of $d = 3$. The above worst case scenarios are not typical of real systems. Nonetheless, even in the worst case, the relative deviation from optimal hardly goes beyond a few percent and the policy remains reasonable even in those situations.

Our final test aims at evaluating the policy in the presence of unequal deadlines, unequal lengths, and unequal weights being assigned to the files. This set consists of 100 simulations in which the values of file lengths, deadlines and weights for each file are assigned by separately sampling a $uniform(1, 10)$ distribution. The graph of figure 7 shows the G values for these randomized parameter values. Again, the results confirm the close match between the experimental results and the optimal values with smaller than 1% deviations.

Although our main goal has been to achieve the minimum average cost in each problem, it is constructive to study the treatment of the individual files by the optimal policy and also, how close the heuristic policy approximates the optimal policy for each file. The first question can be answered by looking at our previous derivations and

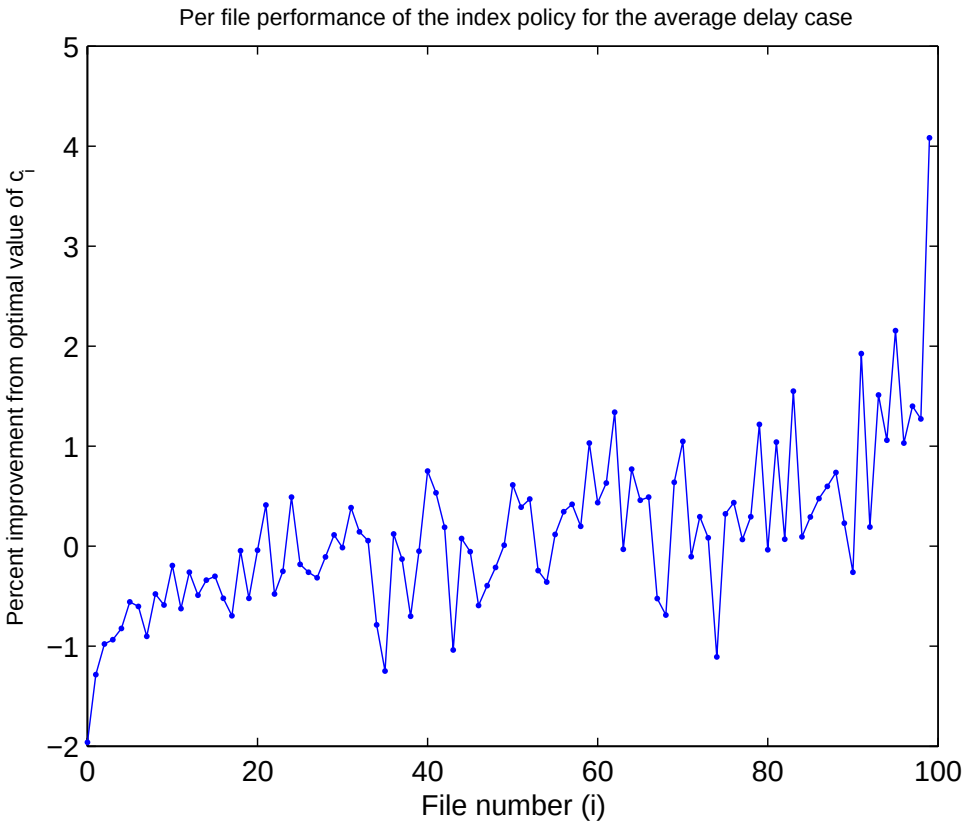

Fig. 8. Relative difference between the experimental results and optimal values of the average delay for each file. Positive values indicate smaller (better) than optimal and vice versa.

plugging in equation (19) in (16) to find all individual average tardiness values as functions of μ. Since the resulting functions are not easy to investigate, we consider the case with all $d_i = 0$. In that case, we have for all i and j,

$$\frac{c_i}{c_j} = \frac{a_i \tau_i}{a_j \tau_j} = \frac{\sqrt{q_j}}{\sqrt{q_i}} \frac{\sqrt{l_i}}{\sqrt{l_j}} \frac{\sqrt{a_i}}{\sqrt{a_j}}.$$

Non-zero d_i values introduce skewness to the above relations and should be mainly investigated separately for any specific scenario.

The second question deals directly with how well our proposed policy approximates the optimal policy. Although at this point we are not able to make a general statement about this subject, we focus on a simplified setting and try to answer this question for that case. In particular, we consider a system with all l_i and a_i values being set to one and $d_i = 0$ for all i i.e., the average delay case. Figure 8 shows the percentage of deviation from the optimal values of average delays for individual files. It can be seen that the index policy favors the files with lower demands by providing them with a smaller-than-optimal average delay. Obviously, this effect results in a larger overall average delay. For the same setting, we can record the individual values of the time differences between all successive broadcasts of each file i and observe how close to τ_i those value are. Figure 9 plots the percentage of change towards better (smaller) values between the average inter-broadcast times for every file i and their optimal τ_i values. As we expect, this graph is similar to the previous experiment and the heuristic policy favors the files with smaller demand. The answer to the above questions in general is beyond the scope of this paper and the subject is still under study.

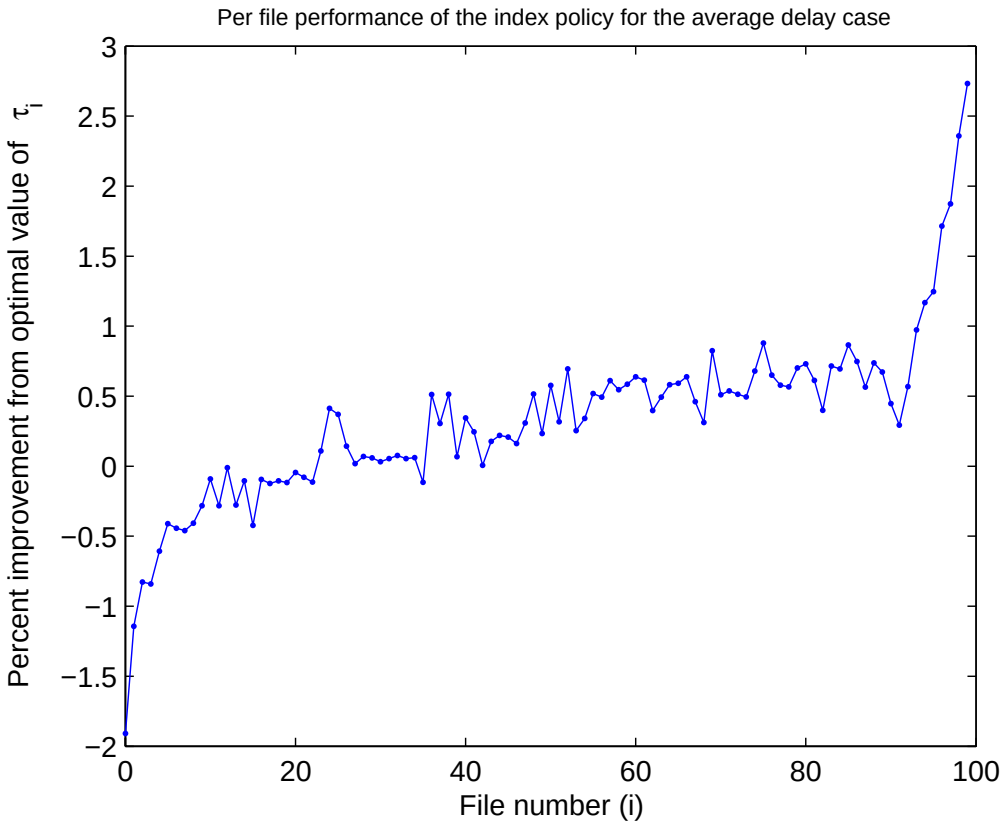

Fig. 9. Relative difference between the experimental results and optimal values of the average inter-broadcast times for each file. Positive values indicate smaller (better) than optimal and vice versa.

We may call our policy a greedy approximation of the ideal policy due to the fact that only the current value of the index function for each file is considered and the file with the largest value is broadcasted. In reality, after the broadcast of a file starts, the index functions of all files and their corresponding costs keep increasing according to their own square-law functions defined by equation 20 for the whole duration of the current broadcast. Theoretically, one may also think of a look-ahead approach that takes this fact into account and come up with a more complicated policy. Although it is not known if such policies may result in a better performance, our experimental results are reasonably close to the optimal values and we do not see any need for studying such policies. Another observation we would like to mention is about the performance of a slightly different index policy derived from the same optimality arguments. Since the optimal values of τ_i are precisely defined by the optimal policy for all files, it is reasonable to think of an index policy that only uses the $t_i - \tau_i$ value as the index function for every file and broadcasts the file with the largest difference. Unfortunately, experimental results showed that this new policy performs substantially worse than our original policy (which can be shown also as $a_i q_i (t_i^2 - \tau_i^2)/l_i$). Therefore, we focused our attention to the original policy throughout this paper.

Overall, all of the above results indicate that the proposed scheduling policy with an $O(N)$ complexity performs very close to optimal at least for the family of cost functions defined by figure 3. Our derivations, however, allow for arbitrary cost functions and we do not expect a drastic reduction in the performance for other typical smooth cost functions.

V. Conclusion

This paper presents a general formulation of the scheduling policy in Push broadcast systems. Our formulation addresses the systems with generalized cost functions and provides index policies for them. We introduced an auxiliary problem and finding the optimal solution for that system enabled us to propose a scheduling policy for the original system. Our results are based on the comparison between the performance of the policy and the theoretical lower bounds. In all of our experiments, even the worst case result was smaller than 2% different from the optimal result. A similar technique is also applicable to Pull systems and this problem is the subject of ongoing research.

Acknowledgment

The authors would like to thank Dr. Samy Abbes for his helpful suggestions and comments about this research.

Appendix I
Optimality of the periodic schedule

Proof: We show that having the time between any kth and $k+2$th broadcasts of the file fixed, the optimal time for the $k+1$th broadcast is in the middle of that interval i.e., we should have

$$t_{k+2} - t_{k+1} = t_{k+1} - t_k. \tag{27}$$

Let's assume that the total length of the $[t_k, t_{k+2}]$ interval is L and the $k+1$th service happens at $t_k + T$ with $T < L$. Due to the Poisson assumption, the expected cost for each of the arriving requests during the $[0, T]$ interval is given by

$$\begin{aligned} W_1 &= \int_0^T C(T-t)\frac{1}{T}dt \\ &= \frac{1}{T}\int_0^T C(t)dt \end{aligned} \tag{28}$$

which in the average delay case will reduce to $T/2$ as expected. Now, if we calculate the average cost for the requests arriving in $[0, L]$, we have

$$\begin{aligned} W(T) &= \frac{T}{L}W_1 + \frac{L-T}{L}W_2 \\ &= \frac{1}{L}\left(\int_0^T C(t)dt + \int_0^{L-T} C(t)dt\right). \end{aligned}$$

A simple differentiation with respect to T shows that the average cost is minimized when

$$W'(T) = C(T) - C(L-T) = 0 \tag{29}$$

with

$$W"(T) = C'(T) + C'(L-T).$$

If $C(t)$ is monotonic increasing, we have $W"(T) > 0$ and equation 29 has a unique solution at $T = \frac{L}{2}$. If $C(t)$ is non-decreasing, $T = \frac{L}{2}$ is still a solution though not necessarily unique. ■

APPENDIX II
GENERAL PROPERTIES OF THE OPTIMAL SOLUTION

Proof: Do to the summation form of the objective function we have

$$\frac{dC}{d\tau_i} = q_i \frac{dc_i}{d\tau_i}.$$

But

$$\begin{aligned}\frac{dc_i}{d\tau_i} &= \frac{d}{d\tau_i}\frac{1}{\tau_i}\int_0^{\tau_i} C_i(t)dt \\ &= -\frac{1}{\tau_i^2}\int_0^{\tau_i} C_i(t)dt + \frac{1}{\tau_i}C_i(\tau_i) \\ &= \frac{1}{\tau_i^2}\left(\tau_i C_i(\tau_i) - \int_0^{\tau_i} C_i(t)dt\right).\end{aligned}$$

If $C_i(t)$ is an increasing function of t, we have

$$\begin{aligned}\int_0^{\tau_i} C_i(t)dt &< \int_0^{\tau_i} C_i(\tau_i)dt \\ &< \tau_i C_i(\tau_i).\end{aligned}$$

Based on the above equations, we have $dc_i/d\tau_i > 0$ and, subsequently, $dC/d\tau_i > 0$. Also, for the constraint function,

$$\frac{d}{d\tau_i}\sum_{i=1}^{N}\frac{l_i}{\tau_i} = -\frac{l_i}{\tau_i^2}.$$

The increasing objective function together with the decreasing constraint function with respect to all variables, forces the optimal point to be at a point where the constraint function meets its upper bound since any slackness can be used to reduce one of the τ_i values which, in turn, leads to a smaller C. Therefore. Again, for $C(t)$ being non-decreasing, the same solution is optimal but not necessarily unique. Therefore, the optimal solution always satisfies

$$\sum_{i=1}^{N}\frac{l_i}{\tau_i^*} = 1$$

■

REFERENCES

[1] J. W. Wong and M. H. Ammar, "Analysis of broadcast delivery in a videotext system," *IEEE Trans. on computers*, Vol. C-34, No. 9, pp863-966, 1985.

[2] S. Acharya, M. Franklin, and S. Zdonik, "Balancing push and pull for data broadcast," *Proc. ACM SIGMOD, Tuscon, Arizona.*, 1997.

[3] J. S. Baras K. Stathatos, N. Roussopoulos, "Adaptive data broadcast in hybrid networks," *The VLDB Journal*, pp. 326–335, 1997.

[4] M.H. Ammar and J.W. Wong, "On the optimality of cyclic transmission in teletext systems," *IEEE Trans. Comm.*, pp. Vol. 35, pp68–73, Jan. 1987.

[5] M.H. Ammar and J.W. Wong, "The desgning of teletext broadcast cycles," *Perf. Eval.*, pp. Vol. 5, No. 4, pp235–242, Nov. 1985.

[6] N. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments," *Wirel. Netw.*, vol. 5, no. 3, pp. 171–182, 1999.

[7] C. Su and L. Tassiulas, "Broadcast scheduling for information distribution," *Proc. of INFOCOM 97*, 1997.

[8] C. J. Su and L. Tassiulas, "Broadcast scheduling for the distribution of information items with unequal length," *Proc. of the 31st Conference on Information Science and Systems (CISS'97)*, Mar. 1997, Baltimore, Maryland.

[9] M.H. Ammar, "Response time in a teletext system: an individual user's perspective," *IEEE Trans. Comm.*, pp. vol. 35, pp1159–1170, Nov. 1985.

[10] A. Bar-Noy, "Optimal broadcasting of two files over an asymmetric channel," *J. Parallel and Distributed Computing*, pp. Vol. 60, pp474–493, 2000.

[11] N. Vaidya and H. Jiang, "Data broadcast in asymmetric wireless environments," *Proc. 1st Int. Wrkshp Sat.-based Inf. Serv.(WOSBIS). NY*, Nov. 1996.

[12] E. W. Zegura M. J. Donahoo, M. H. Ammar, "Multiple-channel multicast scheduling for scalabel bulk-data transport," *INFOCOM'99*, pp847-855, 1999.

[13] Q. Hu, D. L. Lee, and W. Lee, "Dynamic data delivery in wireless communication environments," *Workshop on Mobile Data Access*, pp213-224, Singapore, 1998.

[14] J. Xu, D. Lee, Q. Hu, and W. Lee, "Data broadcast," *in Handbook of Wireless Networks and Mobile Computing, edited by I. Stojmenovic*, John Wiley & Sons Inc., 2002.

[15] M. Raissi-Dehkordi and J. S. Baras, "Broadcast scheduling in information delivery systems," *Proceedings of IEEE GLOBECOM2002*, Nov. 2002, Taipei, Taiwan.

[16] M. Raissi-Dehkordi and J. S. Baras, "Near-optimal scheduling policies for broadcast of files with unequal sizes in satellite systems," *Allerton conference on communications, control, and computing*, Oct. 2003, Allerton, Illinois.

[17] P. Whittle, "Restless bandits: activity allocation in a changing world," *A Celebration of Applied Probability, ed. J. Gani, J. Appl. Prob.*, 25A, pp287-298, 1988.

[18] P. P. Bhattacharya and A.E. Ephremides, "Optimal allocation of a server between two queues with due times," *IEEE trans. Automatic Control*, Vol.36,No.12,pp1417-1423,Dec 1991.

[19] G.K. Zipf, *Human Behaviour and the Principle of Least Effort.*, Addison-Wesley, Cambridge, Massachusetts, 1949.