

# Near-optimal policies for broadcasting files with unequal sizes\*

Majid Raissi-Dehkordi and John S. Baras  
Institute for Systems Research  
University of Maryland  
College Park, MD 20742  
majid@isr.umd.edu, baras@isr.umd.edu

## Abstract

Information broadcasting is an effective method to deliver popular files to a large number of users in wireless and satellite networks. In a previous work, we used a dynamic optimization approach to address the problem of broadcast scheduling for a *pull* system with equal file sizes. In this paper, we address that problem in a more general setting where the file sizes are not equal and have geometric size distributions with possibly different means. The dynamic optimization approach allows us to find a near-optimal scheduling policy, which we use as a benchmark to evaluate a number of other heuristic policies. Also, we modify the resulting policy and apply it to the case with fixed (unequal) file sizes and compare the results with some other well known, as well as new, heuristic policies. Finally, we introduce a low-complexity heuristic policy to be used for practical implementations. The results show that the performance of the new policy is very close to that of the original policy.

## 1 Introduction

Due to the increasing demand for access to information through wireless channels, finding methods for more efficient usage of the bandwidth in satellite and wireless systems has become an important issue. In many systems, popular packages of information are stored in the Network Operation Center (NOC) and the users can retrieve them by sending a request to the NOC and receive the corresponding file via the downlink channel. In wireless and satellite systems, the inherent broadcast capability of the system allows for serving a number of requests, from different users for a single file, simultaneously. This is done by sending a single copy of the requested file on the broadcast channel so that all of the requesting users can receive it at the same time. We call this type of systems the *pull broadcast* or *pull* systems as opposed to the *push* systems where the files are broadcasted irrespective of the instantaneous number of requests for them.

In a *pull* system with a large number of stored files, the server receives all the requests and must schedule the transmissions based on the number of requests for different files. The objective of the scheduling is usually the average waiting time of the requests for all files. Compared to the wealth of research works on the scheduling problem in *push* broadcast systems, fewer works have addressed the scheduling problem in the *pull* systems. The papers by Ammar, Dykeman and Wong[1][2] introduced this problem and presented both heuristic and numerical solutions for special cases. Later, Franklin and Aksoy[3] presented a heuristic policy (LxW) that used the number of requests for each file together with the time since the last broadcast of the file to calculate the index associated with that file. The policy would then chose the file with the largest index value to be broadcasted. Su and Tassiulas[4] introduced another

---

\*Research partially supported by NASA cooperative agreement NCC3-528, by MIPS grant with Hughes Network Systems, and by Lockheed Martin Networking Fellowship all with the Center for Satellite and Hybrid Communication Networks at the University of Maryland at College Park.

index policy (PIP), which used the number of requests for each file together with the request arrival rate for the file to calculate the index associated with that file. Both of these policies performed almost identical to each other and also to the LTWF performance. In [4], a Markov Decision Process(MDP) formulation of the problem was also presented. However, the complex form of the problem prevented them from going very far with that approach. The work in [5] is probably the only one to use a MDP formulation of the problem and find an analytical solution for it. They investigated the problem in a more general framework and derived a near-optimal index policy using the MDP formulation. They also used that policy to propose low complexity heuristic policies that extend the PIP policy to this more general setting. All of the above works on *pull* systems with minimum average delay objective assume equal file sizes. This restriction results in a degradation of the performance when those policies are applied to more realistic systems where the stored files have different sizes.

In this paper we investigate the same problem i.e., optimal scheduling of the broadcasts in a *pull* system in order to minimize the average waiting time of the users. However, we investigate the problem when the file sizes need not be equal. This situation is more general than the fixed-length setting[5] and applies to a larger number of practical situations. To our knowledge, there has not been any previous work, neither heuristic nor analytical, on this more general problem and this work seems to be the first attempt to study it. We propose a MDP formulation of the problem and use the ideas from the bandit problems to propose index-type scheduling policies. We study the problem with random file sizes and later extend the policy to the case with deterministic file sizes as well. In section 2, we present our MDP formulation of the problem and the restless bandit approach for solving it. In section 3, after proving the necessary properties of the system, we find a heuristic index policy for this problem. Section 4 is dedicated to evaluation of the policy and comparing it with some other policies. Due to the lack of previous works on this problem, the well known scheduling policies do not cover our system. Hence, we deviate from our evaluation and first find an experimentally optimized heuristic policy for our problem and compare its results with that of our initial policy.

## 2 Problem formulation

We denote by  $N(> 1)$ , the number of files stored in the system. We also assume that the broadcasts can only start in certain time instants which are equally spaced in time. This periodic setting introduces a time unit that can be set to one without any loss of generality. The file sizes are random variables with Geometric distributions with parameter  $q_i$  for type  $i$  files. If we denote by  $l_i$  the length of file  $i$ , we have

$$P[l_i = n] = q_i(1 - q_i)^{n-1}, n \geq 1, 0 < q_i \leq 1, i = 1, \dots, N. \quad (1)$$

Here we implicitly assume that the sizes are rounded up to the smallest integer multiple of the above time unit. We also allow preemption in the system, i.e. the broadcast of a file can be interrupted by the system, so that another file is broadcasted, and can be resumed at a later time. However, this can only happen at the beginning of every broadcast period. Therefore, every broadcast initiation time  $t = 0, 1, \dots$  is a decision time (and also a possible preemption time). The waiting time of the requests for a file is defined as the time since the arrival of the request until the end of the transmission of the last segment of that file. The new requests for each file which arrive after the beginning of the transmission of the first segment of the file, need to wait till the beginning of the next transmission of the file. We also assume that the system has  $K(1 \leq K < N)$  identical broadcast channels. This *pull* system has complete knowledge about the number of pending requests for each file and based on this information determines the file to broadcast in the next time unit in order to minimize the average waiting time over all users.

The request arrival process for each file  $i; i = 1, \dots, N$  is a discrete-time, stationary, iid process

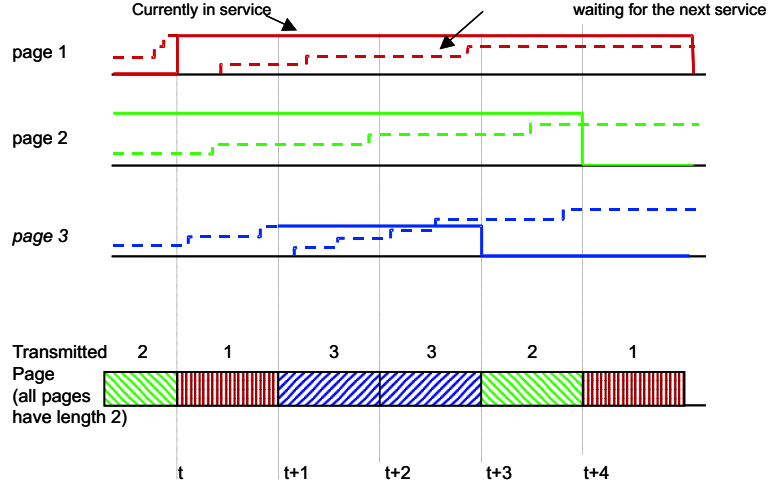


Figure 1: Sample path of a system with three files.

which we show by  $A_i(t)$ ;  $t = 0, 1, \dots$ . We denote by  $p_i(a)$ ;  $a \geq 0$  the pmf of the arrivals during every time unit and show its mean value by  $\lambda_i$ . In this paper, we assume that the time slots are short enough so that the probability of having more than one arrival for *each* file is negligible, though many of our results hold for the general case as well. The state of the system at any time instance  $t$  is  $\mathbf{X}(t) = (X_1(t), Y_1(t), X_2(t), Y_2(t), \dots, X_N(t), Y_N(t))$  where  $X_i(t)$  is the number of requests for file  $i$  at time  $t$  that have received at least one segment of the requested file and  $Y_i(t)$  is the number of requests for the same file which arrived after the broadcast of the first segment of the file and therefore need to wait till the next full broadcast of that file. Each  $(X_i(t), Y_i(t))$ ;  $i = 1, \dots, N$  process is a Markov process with transition probability

$$(X_i(t+1), Y_i(t+1)) = \begin{cases} (0, Y_i(t) + A_i(t)) & w.p. q_i & \text{if } i \in d(t) \\ (X_i(t), Y_i(t) + A_i(t)) & w.p. (1 - q_i) & \text{if } i \in d(t) \\ (X_i(t), Y_i(t) + A_i(t)) & & \text{if } i \notin d(t) \end{cases} \quad (2)$$

if  $X_i(t) > 0$  and

$$(X_i(t+1), Y_i(t+1)) = \begin{cases} (0, A_i(t)) & w.p. q_i & \text{if } i \in d(t) \\ (Y_i(t), A_i(t)) & w.p. (1 - q_i) & \text{if } i \in d(t) \\ (0, Y_i(t) + A_i(t)) & & \text{if } i \notin d(t) \end{cases} \quad (3)$$

if  $X_i(t) = 0$ . Here  $d(t) \subset \{1, \dots, N\}$  is the set containing the indices of the  $K$  files broadcasted at time  $t$ . Figure (1) shows a sample path of the evolution of a system with three files and a single broadcast channel.

The average waiting time over all users is defined by

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

where  $\bar{W}_i$  is the average waiting time for all file  $i$  requests and  $\lambda$  is the total request arrival rate to the system. By Little's law the average waiting time can be written as

$$\bar{W} = \frac{1}{\lambda} \sum_{i=1}^N (\bar{X}_i + \bar{Y}_i). \quad (4)$$

where  $\bar{X}_i$  and  $\bar{Y}_i$  are the average numbers of the requests currently in service or waiting for service in queue  $i$ , respectively. To avoid the difficulties associated with the average cost problems,

instead of minimizing (4), we use the total discounted reward criteria and try to minimize the total discounted expected number of waiting requests defined as

$$J_\beta(\pi) = E \left[ \sum_{t=0}^{\infty} \beta^t \sum_{i=1}^N (X_i(t) + Y_i(t)) \right]. \quad (5)$$

Here  $\pi$  is the scheduling policy resulting in  $J_\beta(\pi)$  and under mild conditions[6]  $(1 - \beta)J_\beta(\pi)$  approaches the optimal value for problem (4) as  $\beta \rightarrow 1$ . Equations (5) and (2), together with the initial condition  $(X(0), Y(0))$ , define the minimization problem

$$J_\beta^*(\pi) = \min_{\pi} E \left[ \sum_{t=0}^{\infty} \beta^t \sum_{i=1}^N (X_i(t) + Y_i(t)) \right]. \quad (6)$$

It can be shown[7] that  $J_\beta(\pi)$  satisfies

$$(1 - \beta)J_\beta(\pi) = E \left[ \sum_{i=1}^N (X_i(0) + Y_i(0)) \right] + \beta E \left[ \sum_{t=0}^{\infty} \beta^t \sum_{i=1}^N A_i(t) \right] - \beta E \left[ \sum_{t=0}^{\infty} \beta^t \sum_{i \in d(t)} q_i(X_i(t) + Y_i(t)I[X_i(t) = 0]) \right].$$

Therefore, since the first two terms of the right-hand side are independent of the policy  $\pi$ , the problem of minimizing  $J_\beta(\pi)$  would be equal to the maximization problem

$$\hat{J}_\beta(\pi) = \max_{\pi} E \left[ \sum_{t=0}^{\infty} \beta^t \sum_{i \in d(t)} q_i(X_i(t) + Y_i(t)I[X_i(t) = 0]) \right]. \quad (7)$$

Our goal is to find near-optimal policies for this maximization problem.

### 3 Derivation of the index policy

Problem (7) is a dynamic programming (DP) problem with decision space  $D = \{d; d \subset \{1, 2, \dots, N\} \text{ \& } |d| = K\}$  and state vector  $s = \{x_1, y_1, \dots, x_N, y_N\}$ . Let us denote by  $S$  the state space of the problem. The expected reward for broadcast of files in  $d \in D$  at any state  $s \in S$  is

$$r(s, d) = \sum_{i \in d} q_i(x_i + y_i I[x_i = 0]).$$

Also, if we show the optimal value function of this problem by  $V(s)$ , then  $V(s)$  satisfies the optimality equation

$$V(s) = \max_{d \in D} \left[ r(s, d) + \beta \sum_{s' \in S} p^d(s, s') V(s') \right] \quad \forall s \in S \quad (8)$$

where  $p^d(s, s')$  is the probability of going from state  $s$  to state  $s'$  with decision  $d$  as defined by equations (2) and (3). In generic terms, this problem is a scheduling problem in a queueing system with  $N$  queues and  $K$  servers with different Geometric service times for different queues. The additional property which distinguishes this problem from the similar well-known scheduling problems [8, 9] is the fact that the servers are of the bulk service type with infinite bulk size.

In this work, we limit our search to non-idling policies. Given the fact that there is no cost

associated with each service, it can be shown that a non-idling optimal policy always exists. Moreover, for practical reasons, we are only interested in *index* policies. An index policy assigns a value (*index*) to each queue and picks the queue(s) with largest index for service. Whittle's formulation of the restless bandit problems[10][11] provides a general framework for finding heuristic index policies for this type of problems where a limited resource should be allocated to a finite number of controllable Markov chains in order to maximize some average or time-discounted reward function. That heuristic policy also benefits from some form of asymptotic optimality [12]. However, its existence and the form and complexity of the index function depends on the properties of the problem at hand. For the current problem, we need to consider the auxiliary single-queue problem defined as follows:

Imagine one of our bulk service queues with arrivals and service times as before. The sub-problem we are interested at is to find the optimal policy that results in the maximum expected value of the discounted reward given a fixed service cost  $\nu$  for each service. The optimal policy is the optimal assignment of active (serving the queue) or passive (leaving the queue idle) actions to every state. More precisely, the objective function is:

$$J_\beta = E \left[ \sum_{t=0}^{\infty} \beta^t R(t) \right]$$

where  $R(t)$  is the reward at time  $t$ , that is

$$R(t) = \begin{cases} x(t) - \nu & w.p. q & if d(t) = 1 \ \& \ x(t) > 0 \\ y(t) - \nu & w.p. q & if d(t) = 1 \ \& \ x(t) = 0 \\ -\nu & w.p. 1 - q & if d(t) = 1 \\ 0 & & if d(t) = 0 \end{cases}$$

where  $d(t)$  is the action at time  $t$  which is 1 if the queue is served and 0 otherwise and  $(x(t), y(t))$  is the state of this system at time  $t$  as defined before.

With the definition of the single-queue sub-problem, the requirements and the index function defined by the restless bandit formulation are as follows. We need to find if the solution to the auxiliary single-queue problem is of the threshold type i.e., there exists a curve defined by a set of points in the state space  $S$  where it is optimal to serve the queue if  $(x(t), y(t))$  falls inside (or below) the curve and leave the queue idle otherwise. The switching curve is of course a function of system parameters including  $\nu$ . If the switching curve is a non-decreasing function of  $\nu$ , the near-optimal index function for each queue  $i = 1, \dots, N$  in the original problem is the value of  $\nu_i$  that places the switching curve over the state  $(x_i, y_i)$ .

For any given  $\lambda$ ,  $\beta$  and  $q$ , The optimal policy can be found by numerically solving the above DP problem using one of the well-known numerical methods[6]. Figure (2) shows one example of the form of the optimal policy with the idle and active regions distinguished. It can be seen that the optimal policy is of the threshold type in both  $x$  and  $y$  directions (except  $x = 0$ ). In fact,

**Property 1** *If  $d(x, y)$  is the decision defined by the optimal policy for state  $(x, y)$  we have*

$$if d(x, y) = 1 \ then \ d(x + i, y) = 1; \ \forall x > 0 \ and \ i > 0; \tag{9}$$

*Proof:* [7].

Although all of our numerical results (e.g. figures 3 and 2) also confirm the threshold property in the  $y$  direction, a general proof for this property proved to be rather difficult and involves considering different situations depending on the parameter values. However, the interested reader is referred to [7] for a detailed discussion of the proof for the properties of the switching curve using the induction over policy iteration method along with other algebraic arguments. These properties basically describe the threshold property in the  $x$  and  $y$  directions in more

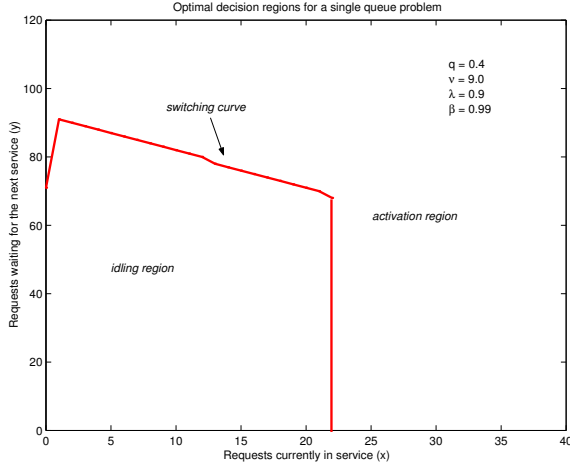


Figure 2: Typical shapes of the idle and active regions for a single queue problem.

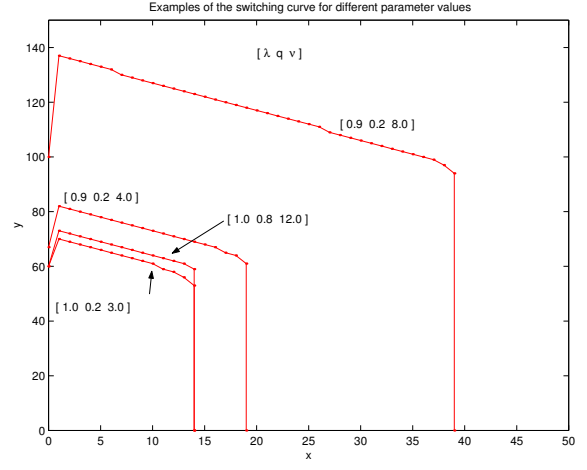


Figure 3: Optimal switching curves for problems with different parameter values.

detail. In fact, in [7], the continuous space approximation of the value function for this problem is used to also find the exact location (within the rounding errors) of the switching curve. It can be shown that the intersection of the curve with the  $x = 0$  axis, shown by  $y_0$  is given by

$$y_0 = \frac{\nu}{q} + \frac{\beta p_1}{1 - \beta} (1 - c^{-y_0}) \quad (10)$$

where  $c = \frac{1 - \beta p_0}{\beta p_1}$ ,  $p_0 = 1 - p_1$  and  $p_1$  is the rate or equivalently, the probability of one arrival. The intersections of the switching curve with  $x = ct$  lines shown by  $y_x$ ;  $x = 1, \dots, x_0$  are given by

$$y_x = y_0 + \frac{1}{\beta a} \left[ \frac{\nu}{q} - x \right] - \frac{a p_1}{q}. \quad (11)$$

where  $a = \frac{q}{1 - \beta(1 - q)}$  and  $x_0 = \lfloor \frac{\nu}{q} \rfloor$ . Also,  $y_x = 0$ ;  $x > x_0$ . For the switching curve defined with the above equations the following property holds:

**Property 2** all  $y_x$ ;  $x = 0, \dots, x_0$  values are increasing functions of  $\nu$ .

*Proof:* [7].

Since  $x_0$  is also a non-decreasing function of  $\nu$ , the idling region defined by the above values is also a non-decreasing function of  $\nu$ . The above equations also allow us to find the index function for any  $(x, y)$  state. As mentioned before, the index function at any state  $(x, y)$  is by definition the amount of service cost  $\nu$  that puts the point  $(x, y)$  on the switching curve. for  $x \leq x_0$ , the  $y_0$  value satisfies

$$y + \frac{1}{\beta a} \left[ \frac{\beta p_1}{1 - \beta} + x \right] + \frac{a p_1}{q} = y_0 \left( 1 + \frac{1}{\beta a} \right) + \frac{p_1}{a(1 - \beta)} c^{-y_0}. \quad (12)$$

Having found the value of  $y_0$ , the corresponding  $\nu$  is

$$\nu = q y_0 - \frac{q \beta p_1}{1 - \beta} (1 - c^{-y_0}). \quad (13)$$

If the resulting  $\nu$  turns out to be smaller than  $qx$ , then  $x$  is on the right border of the idling region, i.e.  $\nu = qx$ . For  $x = 0$  case, the available  $y$  is in fact the  $y_0$  value and  $\nu$  is directly calculated from equation (13).

Having found the index function, the near-optimal scheduling policy is to calculate the index  $\nu_i(x_i(t), y_i(t))$  associated with each queue  $i = 1, \dots, N$  at any decision instance  $t$ , and broadcast the  $K$  queues with the largest index values. In the next section, we compare the results of this policy with those of other well-known policies through simulation studies.

## 4 Results

Unfortunately, to our knowledge, the broadcast scheduling problem with unequal file sizes has not been addressed before. Therefore, we do not have any immediate rival policy readily available for comparison. However, based on previous experiences, we chose a number of well-known policies used in simpler broadcast systems for comparison. Also, we suggested a low complexity index function with three parameters and experimentally optimized the parameter values via a large number of simulations. We then used this policy as one of our candidate policies and evaluated its performance with respect to our initial policy and also to the other candidate policies. In all experiments, we extended our original index function to general arrival rates by replacing the light traffic rate  $p_1$  in the equations with the actual rate  $\lambda_i$  for each queue.

### 4.1 Candidate policies

We compared our policy, which we named NOP(Near-Optimal Policy), with a number of other policies. We set up a system with 50 files and simulated it under different settings with each policy. Other than the choice of the scheduling policy, every experiment had two other sets of parameters namely, the average sizes of the files  $1/q_i$ ;  $i = 1, \dots, N$  and, the total request arrival rate of the system  $\lambda$ . In all experiments, we used the Zipf law to assign the individual request arrival rates  $\lambda_i$  to each queue  $i$  given the total request arrival rate  $\lambda$  of the system. In other words, for  $j > i$ , we have  $\frac{\lambda_i}{\lambda_j} = \frac{j}{i}$  and  $\sum_{i=1}^N \lambda_i = \lambda$ . In order to investigate the effect of the choice of the average file sizes on the performance of the policy, we performed our experiments for two different choices. In one set of experiments, we used  $q_i = 1/i$  and in the other set  $q_i = 1/(N - 1 - i)$  for  $i = 1, \dots, N$ . In other words, the first set assigns the largest size (on average) to the least popular file ( $i = N$ ), and the smallest size to the most popular file ( $i = 1$ ). The second set uses the inverse assignment so that the most popular file also is the longest file (on average). In the following, we use the terms *increasing assignment* and *decreasing assignment* for these two methods, respectively. The policies that were used in the final set of experiments are:

- **NOP**: The index policy derived in this paper.
- **EPIP**: Our extension of the original PIP policy introduced in [4] extended for the new two dimensional setting

$$\nu_i = \frac{x_i + c_y y_i}{\sqrt{\lambda_i}}$$

- **EMRF**: Our extension of Maximum-Request-First index defined as

$$\nu_i = (x_i + c_y y_i)$$

- **FCFS**: First-Come-First-Serve index defined as the current waiting time of the oldest request in each queue.
- **HP2**: Heuristic policy defined as

$$\nu_i = \frac{(x_i + c_y y_i) q_i^{\alpha q}}{\lambda_i^{\alpha \lambda}}$$

Although the FCFS is a well-known policy in the queueing systems, our experiments showed that it performs significantly worse than the other policies under consideration. Therefore, we do not include its results in our graphs and concentrate on the closer behavior of the other four policies. In the above equations,  $c_y$  is an additional weight parameter to allow for more

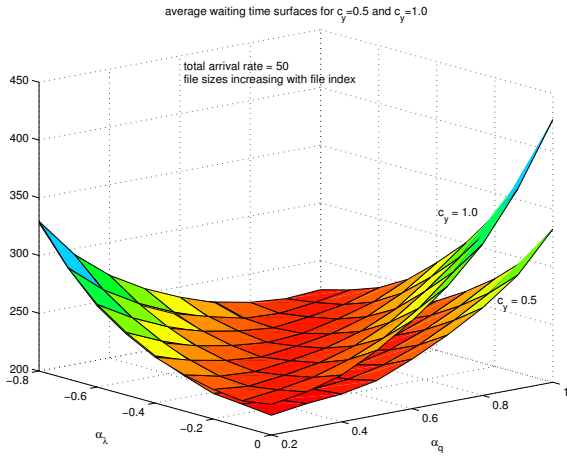


Figure 4: Effect of  $c_y$  on the performance of the heuristic policy.

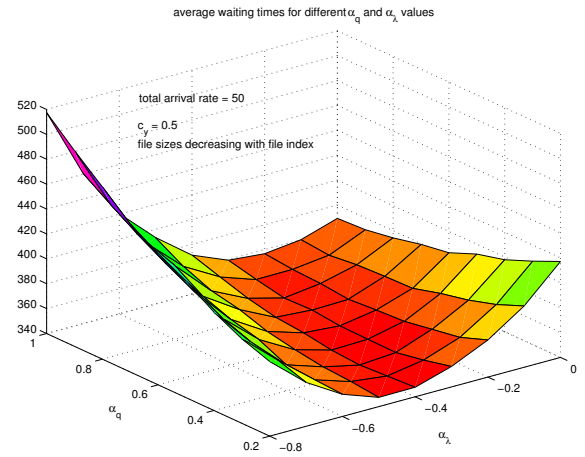


Figure 5: Sample performance surface for decreasing average file size assignment.

flexibility in the index functions. We optimized this value for PIP and MRF policies by running a number of simulations with different arrival rates (5, 10, 20, 50, 100, 150, 200) and different  $c_y$  values (0.1, 0.2, ..., 1.0) and finding the value that resulted in the smallest average delay for each policy. Interestingly, for both policies  $c_y = 0.5$  gave the best results.

The heuristic policy HP2 involved a larger number of experiments to tune its three parameters. In our experiments, for each of the two methods for assignment of file sizes, we ran experiments for all combinations of  $\alpha_q \in \{0.0, \dots, 1.0\}$ ,  $\alpha_\lambda \in \{0.0, \dots, 1.0\}$  and  $c_y \in \{0.1, \dots, 1.0\}$ . The experiments were performed for two choices of the total request arrival rates namely  $\lambda = 50$  and  $\lambda = 150$ . Figure (4) shows an example of the average waiting time surfaces for different  $\alpha_q$  and  $\alpha_\lambda$  values and  $c_y = 0.5$  and  $c_y = 1.0$ . In this figure, for any choice of  $\alpha_q$  and  $\alpha_\lambda$ , the average waiting time values for  $c_y = 0.5$  is always smaller than that of the  $c_y = 1.0$ . In all of our experiments, regardless of the choice of the rate, file size assignment and the two exponent values,  $c_y = 0.5$  always gave the best results. Also, in all of the experiments, regardless of the values for the rate, file size assignment and  $c_y$ , the choice of  $\alpha_q = 0.5$  and  $\alpha_\lambda = 0.4$  resulted in the minimum average waiting time among other choices or a value very close to the minimum (figures 5 and 4). We remind that our HP2 policy in its general form includes the EPIP and EMRF policies and after finding the "optimal" parameter values for HP2, we always expect HP2 to outperform these two policies. However, we still include the results for these two policies as well-known policies that may be used by other people.

## 4.2 Performance results

The experiments were performed under seven choices of the total request arrival rates namely,  $\lambda = 5, 10, 20, 50, 100, 150, 200$ . Figure (6) shows the results obtained from all policies for the  $q_i = 1/i$ ;  $i = 1, \dots, N$  file size assignment. The results clearly show that the HP2 and NOP policies perform almost identically. Also, although EPIP results in larger waiting time, it still performs very close to the other two policies. However, EMRF results in a significantly larger waiting time. Figure (7) shows the results for the case with  $q_i = 1/(N - 1 - i)$ ;  $i = 1, \dots, N$ . Although HP2 and NOP again perform almost identical, EPIP results in a significantly larger waiting time for this case. However, EMRF comes closer in performance to the optimal policies. In general, we conclude that the original NOP policy is indeed a near-optimal policy since it performs identical to the experimentally optimized policy HP2. The analytical procedure for finding the NOP, gives it the flexibility to be modified for other variations of the problem. On the other hand, for practical situations that match our experimental settings, HP2 can be used as a low-complexity alternative to NOP.



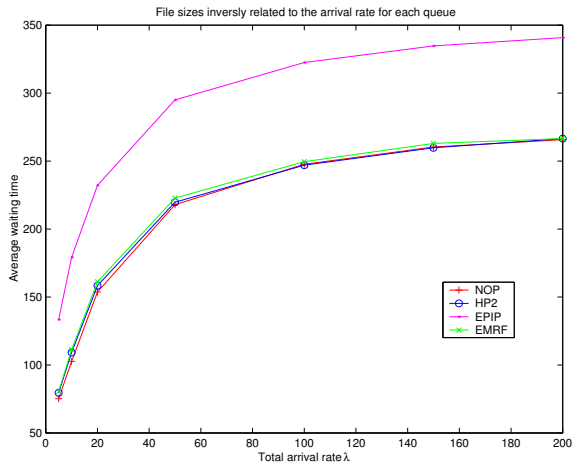


Figure 6: Performances of the policies for increasing average file size assignment.

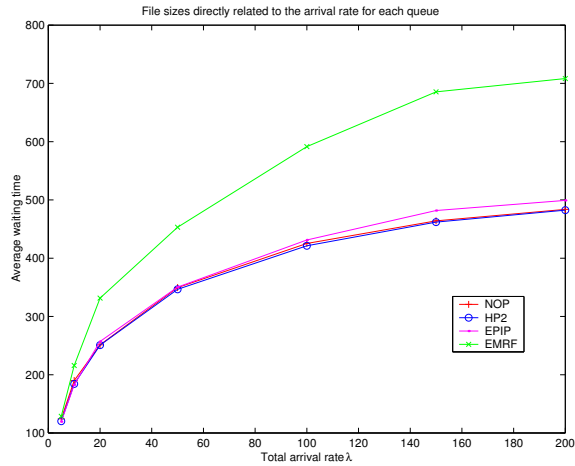


Figure 7: Performances of the policies for decreasing average file size assignment.

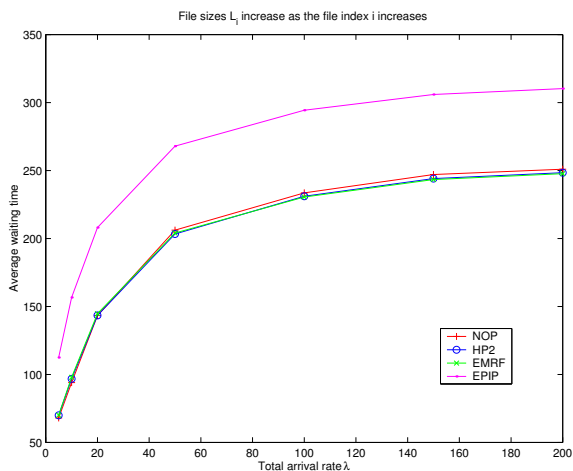


Figure 8: Performances of the policies for increasing file size assignment.

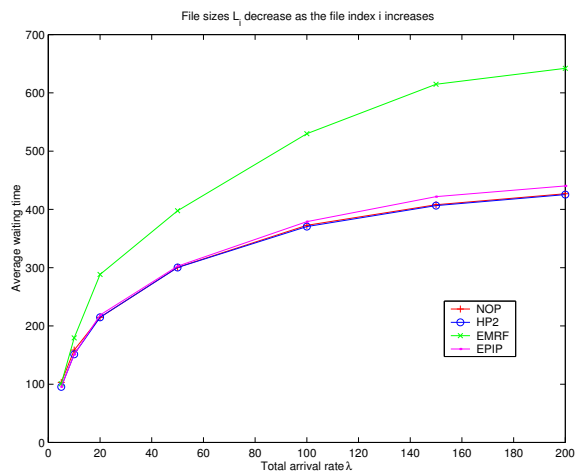


Figure 9: Performances of the policies for decreasing file size assignment.

### 4.3 Fixed file sizes

In some broadcast systems, the files to be broadcasted are locally stored in the system and therefore the system knows their exact sizes. Despite the failure of our approach for this case, the resulting index policy can still be applied to these system as a candidate scheduling policy. Also, all other policies defined above can be defined for this case by replacing the average file size value  $1/q_i$  with its exact value  $L_i$  for each file. As a preliminary investigation, we applied the same four policies namely, NOP, HP2, EPIP and, EMRF to similar broadcast systems but with deterministic file sizes which exactly matched the average size values in the previous cases. Figures 8 and 9 show the results for two choices of the assignment of sizes to the files. The results are basically the same as our previous results for the Geometric file size case. These results suggest that both HP2 and NOP policies might also be near-optimal policies for the deterministic case. However, more investigations and experiments are needed for a better evaluation of this policy.

## 5 Conclusion

In this paper, we used the restless bandit problem formulation to address the problem of optimal scheduling in broadcast systems with random file lengths. We showed that the problem satisfies the requirements for the existence of a near-optimal index policy and derived an equation for the index function for the light traffic regime and extended it for use in moderate traffic cases and fixed file size situations. At the same time, we chose several well-known, as well as new, heuristic policies and tried to optimize them for use in our experiments. All of the results strongly suggest that our policy outperforms the other policies. Also, one of our heuristic policies proved to perform as good as the original policy in all experiments and can be used as a low complexity policy for practical applications. The results suggest that the optimization approach can serve as an effective method for finding near-optimal policies for different variations of this problem with different cost structures or when distinct weights are assigned to different files.

## 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] H. D. Dykeman, M. H. Ammar, and J. W. Wong, "Scheduling algorithms for videotex systems under broadcast delivery," *IEEE Int. Conf. on Comm. ICC86*, Vol. 3, pp1847-51, 1986.
- [3] D. Aksoy and M. Franklin, "Scheduling for large-scale on-demand data broadcasting," *Proc. INFOCOM 98*, Vol. 2, pp651-9, 1998.
- [4] C. Su and L. Tassiulas, "Broadcast scheduling for information distribution," *Proc. of INFOCOM 97*, 1997.
- [5] Majid Raissi-Dehkordi and John S. Baras, "Broadcast scheduling in information delivery systems," *Proceedings of IEEE GLOBECOM2002*, Nov. 2002, Taipei, Taiwan.
- [6] M. Putterman, *Markov Decision Processes : Discrete Stochastic Dynamic Programming*, Wiley, New York., 1994.
- [7] Majid Raissi-Dehkordi, "Broadcast scheduling in information delivery networks," *Doctoral dissertation, Department of Electrical and Computer Engineering*, University of Maryland at College Park, 2002.
- [8] P. Varaiya, J. Walrand, and C. Buyukkoc, "Extensions of the multi-armed bandit problem," *IEEE Transactions on Automatic Control AC-30*, pp426-439, 1985.
- [9] J. Baras, D. Ma, and A. Makowski, "K competing queues with geometric requirements and linear costs: the c-rule is always optimal," *J. Systems Control Lett.*, Vol. 6, pp173-180, 1985.
- [10] 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.
- [11] Jose Nino-Mora, "Restless bandits, partial conservation laws and indexability," <http://www.econ.upf.es/ninomora/>.
- [12] Richard R. Weber and Gideon Weiss, "On an index policy for restless bandits," *J. Appl. Prob.*, Vol. 27, pp637-648, 1990.