Broadcast scheduling in information delivery systems

Majid Raissi-Dehkordi, John S. Baras
Institute for Systems Research
University of Maryland at College Park

Abstract—**Information broadcasting is an effective method to deliver popular information packages to a large number of users in wireless and satellite networks. In this paper, we address the problem of broadcast scheduling in the *pull* environment and try to solve this problem by formulating it as a dynamic optimization problem. This approach allows us to find a near-optimal scheduling policy, which as such, we use as a benchmark to evaluate a number of other existing heuristic policies. Also, in addition to providing a solution for the usual case with Poisson arrivals and equal priority pages, our approach enables us to address the extended versions of this problem with other arrival processes and with distinct weights assigned to different pages.**

I. Introduction

MANY data communication applications are inherently asymmetric. That is, there are a few information sources and a large number of users, and the volume of data transferred from the sources to the users is much larger than that in the reverse direction and we use the term *data delivery* to refer to this specific type of applications. Some data delivery applications have become increasingly popular in recent years. For example, many cellular phones or wireless hand-held devices are currently capable of receiving periodic updates of information like news, weather, traffic or stocks quotes from the air and the number of these applications is expected to increase with the growth of the Mobile Computing field. The WWW traffic can also be regarded as a data delivery application particularly in the networks with caching. One of the major issues in the design of a data delivery system is its scalability. Generally, satellite and wireless environments due to their inherent broadcast capability are the perfect media for highly scalable data delivery systems. The two main architectures for broadcast delivery are the one-way(or *Push*) and the two-way(or *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 able to achieve a better 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], [2], [3]. The main problem with both of the above broadcast methods is the scheduling of data transmission. As we will mention in the next section, the problem of scheduling in a *push* system is solved to a large extent. However, to our knowledge, the problem of finding the optimal broadcast scheduling policy for a *pull* system apparently has not been solved yet.

Based on the nature of the applications supported by a data delivery system, different performance metrics can be used to evaluate the performance of the system. In this work, we try to minimize the *weighted* average waiting time of the users to allow some flexibility in assigning *soft* priorities to the packages.

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.

This paper addresses the scheduling problem in a *pull* system. it aims to find the *optimal* (with respect to the weighted average waiting time) scheduling policy and also provide a benchmark for evaluating current and possibly future heuristic algorithms. We approached the scheduling problem from a dynamic optimization point of view. This formulation is similar to the formulation in [4] and [5] but instead of using numerical methods for extremely simplified versions of the problem or using this formulation to find a few properties of the unknown optimal policy, our goal is to reach an analytical solution and present an index policy through optimization arguments. Using the *Restless Bandit*[6] formulation, our approach naturally addresses the systems with multiple broadcast channels, or prioritized pages and also provides guidelines for the case with unequal page sizes.

This paper is organized as follows. Section II addresses the motivation of our work. There we review the current results for the problem of broadcast scheduling mainly in *pull* systems, the subject of our work, and to some extent in *push* systems. In section III the mathematical formulation of the problem as a dynamic optimization problem is presented. Section IV describes our approach for solving the optimization problem and contains the main results of this work. Finally in section V we compare the performance of our algorithm with some of the current well-known algorithms in this field and present a discussion followed by some concluding remarks.

II. Related work

The series of works by Ammar and Wong are probably the first papers addressing the broadcast scheduling problem in detail. In [7], [8], they consider various aspects of the *push* systems by analyzing the problems associated with a Teletext system. They derive the theoretical lower bound for the average waiting time of the users of a Teletext system and showed that the optimal scheduling policy is of the cyclic type. They also presented a heuristic algorithm to design the broadcast cycle based on the arrival rates. Vaidya and Hameed [9], [10] extended the so called square root formula to cover *push* systems with unequal page sizes and also considered the systems with multiple broadcast channels. There are also a number of other works about *push* systems [11], [12], [13], [5] where all of them address the scheduling problem for different variations of a *push* system.

Despite the wealth of resources about the *push* systems, the number of works addressing the *pull* broadcast systems is very limited. However, none of those papers(except [4], to our knowledge) have tried to find the *optimal* scheduling policy and most of them have suggested heuristic algorithms which despite their good performances in some cases [14], [5], do not contain any notion of optimality. In [4], the problem of finding the optimal scheduling policy for a *pull* system is formulated as a Dynamic Programming(DP) problem. This work might be the first attempt for an analytical approach to the *pull* scheduling problem. However, the question of finding the optimal policy still remains unanswered. In [1] a number of heuristic policies for a *pull* system are proposed and their resulting average waiting times are compared. In [5], an index policy called PIP was introduced and after experimental tuning of the parameter of that function for the case with Zipf distribution of the arrival rates, it resulted in very satisfying results in a number of experiments. The work by Aksoy and Franklin [14] proposes

0-7803-7632-3/02/$17.00 ©2002 IEEE

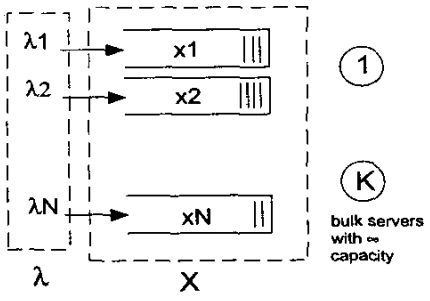

Fig. 1. The pull type broadcast as a queuing system.

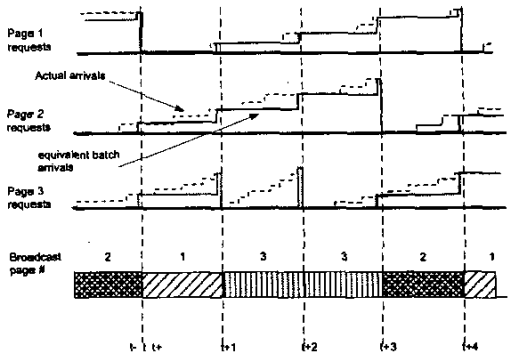

Fig. 2. Sample path of a system with three pages.

another index policy named *RxW* and reports a performance comparable to PIP in different experiments. The two above works are probably the best known scheduling methods for a *pull* system. However, the distance between their performances and that of an optimal policy still remains unknown. From another point of view, all of the above works only consider the case where all pages are of equal importance and have equal sizes and do not apply to cases like cache broadcasting where the pages can have unequal lengths. In the following chapters we present our formulation of the problem and the solution to it.

III. Problem Formulation

In this section a mathematical formulation for the *pull* broadcast scheduling is presented. We denote by $N(> 1)$, the number of information packages stored in the system. In this work we analyze the case where all packages have equal sizes. This assumption is also made in [5], [4], [14] and most of the other works on this subject and is a reasonable assumption for many applications. Throughout this paper, we will use the terms *page* and *information package* interchangeably to simplify the notation and also to stress the equal sizes of the packages. The fixed page size assumption naturally introduces a time unit that is equal to the time required to broadcast a page on a channel and it can be set to one without loss of generality. All of the broadcast times therefore, start at integer times denoted by t; $t = 0, 1, \ldots$.

Here we assume that the system has $K(1 \leq K < N)$ identical broadcast channels. In a *pull* broadcast system, the system receives the requests for all packages from the users and based on this information the scheduler decides which pages to transmit in the next time unit in order to minimize the average waiting time over all users.

For the systems with a large number of users it is reasonable to assume that the requests for each page i; $i = 1, \ldots, N$ arrive as a Poisson process and denote by λ_it he rate of that process. The waiting time for every request is the time since the arrival of the request to the system until the beginning of the broadcast of the requested page. Due to the Poisson assumption for the request arrival process we can assume that the requests for every page i arrive at discrete time instants t as batches of random size having $Poisson(\lambda_i)$ distributions and ignore the residual waiting times without loss of generality. The system therefore, can be shown by a system of N queues where each queue corresponds to one of the packages and holds all the pending requests for that package, and K servers as in figure 1. Due to the broadcast nature of the system, the queues are of the bulk service type [15] with infinite bulk size i.e. the requests waiting in a queue will be served altogether once the queue is serviced. The state of this system at each time t is shown by $X(t) = (x_1(t), x_2(t), \ldots, x_N(t))$ where $x_i(t)$ is the number of pending requests for page i at time t. Each $x_i(t)$; $i = 1, \ldots, N$ is a Markov process with transition probability

$$P_i^{d(t)}(x_i(t) = x_i^1, x_i(t+1) = x_i^2) =$$

$$\begin{cases} p_i(A_i(t)) & if\ x_i^2 = x_i^1 + A_i(t) - x_i^1 1(\mathrm{i} \in \mathrm{d(t)}) \\ 0 & otherwise \end{cases} \quad (1)$$

where $A_i(t)$; $i = 1, \ldots, N$ is the number of new requests for page i during the time interval $[t, t+1)$ and $d(t) \subset \{1, \ldots, N\}$ is the set containing the indices of the K pages broadcast at time t. Figure 2 shows a sample path of the evolution of a system with three pages and a single broadcast channel.

The weighted average waiting time over all users is defined by

$$\bar{W} = \sum_{i=1}^{N} \frac{c_i \lambda_i}{\lambda} \bar{W}_i \quad (2)$$

where $\bar{W}_i$ is the average waiting time for all page i requests and λ is the total request arrival rate to the system. The c_i coefficients are the weights associated with the pages to allow more flexibility in assigning soft priorities to the pages. Due to the discrete-time nature of the system, and to avoid technical difficulties associated with the DP problems with average reward criteria, instead of minimizing (2), we use the policy-dependent total discounted reward criteria. Using Little's law, the problem becomes to minimize the total discounted expected waiting time defined as

$$J_\beta(\pi) = E[\sum_{t=0}^{\infty} \beta^t \sum_{i=1}^{N} c_i x_i(t^+)] \quad (3)$$

where

$$x_i(t^+) = \begin{cases} 0 & if\ page\ i\ is\ broadcast\ at\ time\ t \\ x_i(t) & otherwise \end{cases} \quad (4)$$

and π is the scheduling policy resulting in $J_\beta(\pi)$. Equations (3) and (1), together with the initial condition $X(0)$, define the minimization problem

$$J_\beta^*(\pi) = \min_\pi E[\sum_{t=0}^{\infty} \beta^t \sum_{i=1}^{N} c_i x_i(t^+)]. \quad (5)$$

Which can be shown to be equal to the maximization problem

$$\hat{J}_\beta^*(\pi) = \max_\pi E[\sum_{t=0}^{\infty} \beta^t \sum_{i \in d(t)} c_i x_i(t)]. \quad (6)$$

To facilitate the analysis we assume that the state space of each queue i; $i = 1, 2, \ldots, N$ is a finite set S_i and denote the state space of the system by $S = S_1 \times S_2 \times \ldots \times S_N$. This problem is in fact a DP

problem with decision space $D = \{d;\ d \subset \{1, 2, \ldots, N\}\ and\ |d| = K\}$ where $|d|$ is the cardinality of set d. The decision space D is in fact the set of all possible K tuples of the indices 1 through N. The reward function for broadcast of pages in $d \in D$ at state $s = \{x_1, \ldots, x_N\} \in S$ is

$$r(s, d) = \sum_{l \in d} c_l x_l. \tag{7}$$

This maximization problem is the problem we will address in the sequel to find a non-idling, stationary optimal policy for the *pull* broadcast environment and its derivation completes the formulation of the problem. What we are specially interested in is an index-type policy where there is an index associated with each queue at every instant of time and the optimal decision is to service the queue(s) with the largest index value(s). If the index for each queue only depends on the state of that queue, the computation load for every decision would be of order N which is important from a practical point of view for systems with a large number of stored pages.

IV. Solution of the Optimization Problem

The discrete time nature of our formulation allows us to represent each queue as a controllable discrete time Markov chain. This set up is quite similar to the family of *Bandit* problems introduced in a number of papers by different researchers [16], [17], [18]. In the basic *Multi-armed Bandit* problem there are N independent controllable Markov chains(called projects) and at each instant of time only one of the projects can be activated. With the activation of project $i;\ i = 1, 2, \ldots, N$, a stationary reward of $r = r_i(x_i)$ is achieved, where x_i is the state of project i, and the project $i$cha ngs its state according to its transition probability rule. The passive projects neither produce any rewards nor change their states and the goal is to maximize the expected discounted sum of the rewards. Gittins [19] showed that the optimal policy is of the index type and the index for each project is independent of other projects. The main restriction of the Multi-armed Bandit problem is the requirement that the passive projects do not change their states which is obviously not the case for our system. We therefore use what Whittle [6] introduced as an extension to this problem named the *Restless Bandit* problem that allows the passive projects to produce rewards and change their states too. Unfortunately, with this generalization, the existence of an index-type solution is no longer guaranteed. However, as Whittle showed, in some cases an index-type solution can be found for a relaxed version of this problem that results into reasonable conclusions about the optimal policy for the original problem. In the following we briefly explain the application of this approach to our problem and refer the reader to [6], [20] for more detailed information.

A. Restless Bandit approach

The Linear Programming(LP) formulation of the DP problems [21] and the additive form of the reward in our problem allows us to convert problem (6) into the (dual) LP problem

$$Maximize \sum_{i=1}^{N} \left[\sum_{s \in S_i} r_i(s) z_i(s, 1) \right] \tag{8}$$

subject to

$$\sum_{d \in \{0,1\}} z_i(s', d) - \sum_{s \in S_i} \sum_{d \in \{0,1\}} \beta p_i^d(s, s') z_i(s, d) = \alpha_i(s') \tag{9}$$

for $i = 1, \ldots, N$ and $s' \in S_i$. Here, $\alpha_i(.)$ is the initial probability distribution of the states, $r_i(s)$ is the reward for activating project i while in state s, and $z_i(s, 1)$ is the discounted expected value of the number of times queue i is served while at state s.

An additional constraint implicit to this scheduling problem is that at any time t, exactly K queues should be served. Whittle's relaxation assumes that instead of having exactly K projects activated at any time, only the time average of the number of activated projects be equal to K. Using the Lagrangean Relaxation [22] method and some additional arguments[20], this maximization problem can be broken into N independent maximization problems as

$$Maximize \sum_{s \in S_i} (r_i(s) - \nu) z_i(s, 1) \tag{10}$$

subject to constraint (9) for $i = 1, 2, \ldots, N$ where ν is the Lagrange multiplier of the problem.

The solution to problem (10) is a function of the parameter ν and is an upper bound to the solution of problem (8) with the new relaxed constraint. However, for a specific value ν^* the solutions to both problems will be equal. If ν^* was known, the problem would become finding the optimal policies for each of the N problems in (10) independently where for each queue the reward for serving the queue at state x is $cx - \nu^*$. We have shown [23] that the solution to this single queue problem is of the threshold type that is, it is optimal to leave the queue idle for the set of states $\{0, 1, \ldots, x_i\}$(for some (ν^*)) and to serve the queue otherwise. It can be also shown that the idling set monotonically increases from $\emptyset$ to S_i as the service cost ν^* is increased from $-\infty$ to ∞. The solution to the N queue problem is therefore to service those queues with their states $x_i;\ i = 1, \ldots, N$ larger than their corresponding $x_i(\nu^*)$ thresholds and leave other queues idle. Alternatively, for every state $x_i \in S_i$ of each queue, we can find a value $\nu_i(x_i)$ as the service cost where it is optimal to, leave that queue idle for states $\{0, 1, \ldots, x_i - 1\}$, service the queue for states $\{x_i + 1, \ldots\}$ and equally optimal to serve or not serve at state x_i. Therefore the optimal policy for (10) can be rephrased as: service the queues with $\nu_i(x_i) > \nu^*$ and leave the other queues idle. The above monotonicity property for the queues is in fact the condition for indexability of the projects. Based on this result, a meaningful heuristic for the optimal solution to the original problem with the hard constraint(exactly K queues active at every time) is to find the index $\nu_i(x_i)$ associated with each queue and serve the queues with K largest values of the index. This policy requires a method for calculation of the index function which will be explained below.

B. Calculation of the index function

Calculation of the index function involves finding the proper value of the service cost $\nu(x)$ for a discrete-time queue with infinite bulk service and Poisson arrivals so that it is optimal for the queue to remain idle for states smaller than x, to be serviced for states larger than x and indifferent for x. The optimality is of course with respect to maximum total discounted reward obtained given that the reward from serving the queue at any state $x \in S$ is equal to cx.

Assuming that the value of $\nu(x)$ is known, the value function $V(.)$ of the optimal policy satisfies the set of optimality equations

$$\begin{aligned} V(0) &= \beta \sum_{i=0}^{\infty} p(i) V(0 + i) \\ &\vdots \\ V(x) &= \beta \sum_{i=0}^{\infty} p(i) V(x + i) \\ V(x + 1) &= -\nu + cx + 1c + \beta \sum_{i=0}^{\infty} p(i) V(i) \end{aligned} \tag{11}$$

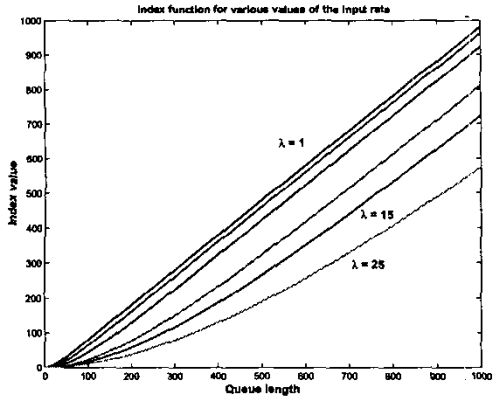

Fig. 3. The form of the index function for Poisson arrivals with different rates.

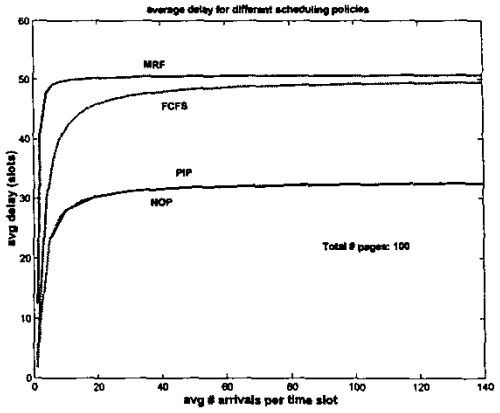

Fig. 4. Comparison of the total average waiting time for different scheduling policies with the distribution of the arrival rates having a Zipf distribution.

where $V(x)$ is the expected reward of the optimal policy given the initial state x. Also, $p(i)$ is the probability of i for a Poisson(λ) distribution. The critical value of $\nu(x)$, by definition, adds another equation to the above system

$$V(x) = -\nu + cx + \beta \sum_{i=0}^{\infty} p(i)V(i).$$

Due to the special form of the $V(.)$ function we have

$$V(x+i) = V(x) + ci; \quad i = 0, 1, \ldots$$

Therefore, the number of unknowns can be reduced to $x+2$ i.e. $V(0), \ldots, V(x), \nu(x)$. The solution to this set of equations can not be expressed in a closed form expression. However, it can be shown [23] that the value of $\nu(x)$ can be calculated via recursive calculations involving the value of $\nu(x-1)$. Figure 3 shows the index function for several values of the input rate and with $c = 1$.

Some thoughts about the form of the above equations reveal an important property of the index function. If we consider the above equations with $c = 1$ and assume that $V_1(.)$ and $\nu_1(.)$ are the solutions of that system, then it is not difficult to show that $V_c(.) = c * V_1(.)$and $\nu_c(.) = c * \nu_1(.)$ are the solutions of the general system with arbitrary c value, hence:

Property 1: If $\nu_c(x)$ is the index function for a bulk service queue with the reward function at state x defined as $r(x) = cx$, we have $\nu_c(x) = c\nu_1(x)$; $x = 0, 1, \ldots$.

This property shows that the index function scales linearly with the weight factor c therefore, we can also extend the definition of the PIP index to cover the weighted case by multiplying the index by c, the original paper on PIP did not give a recipe for calculating the index in this general case.

Another constructive observation is to investigate the shape of the index function in light traffic. In the light traffic regime, the probability of having more than one arrival during any time interval of unit length is negligible. For a $Poisson(\lambda)$ distribution with a small value of λ, a closed form representation of the index function can be found which is

$$\nu(x) = x + \frac{\beta\lambda}{1-\beta}\left[\left(\frac{\beta\lambda}{1-\beta+\beta\lambda}\right)^x - 1\right]. \qquad (12)$$

Although the above formula is meaningful only for $\lambda < 1$ values, we calculated the function for larger values of λ to find the behavior of this index function compared to the exact solution and found reasonably close results for the range of up to roughly $10-15$ requests per period.

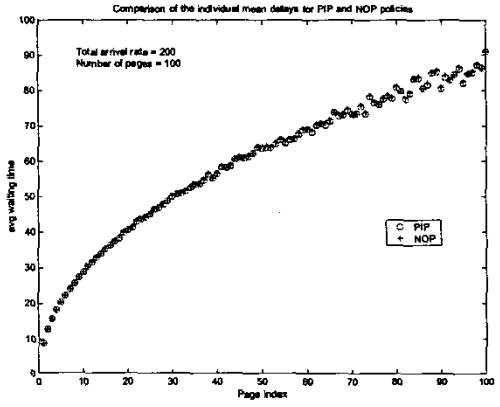

Fig. 5. Average waiting times for the requests for each of the 400 pages under different policies.

V. Results

We tried to evaluate the performance of our policy through several experiments with a single-channel system with 100 pages and used different scheduling policies to schedule the broadcasts. During the experiments, we changed the total request arrival rate λ from small to large values but kept the distribution of the (normalized) individual arrival rates to be according to a Zipf distribution($\theta = 1$). Figure 4 shows the resulting average waiting times for the FCFS, MRF, PIP and our policy which we call NOP(Near-Optimal Policy) for notational convenience. The results show that the performance of PIP is almost the same as the performance of our policy.

Figure 5 shows the individual average waiting times experienced by the requests for each page under PIP and NOP policies for a fixed arrival rate in a system with 100 pages. The close matching of the two results suggests that the PIP policy is probably an approximate version of the optimal policy. Since PIP is optimized [5] for a Zipf distribution of the arrival rates, we compared the performances of PIP and

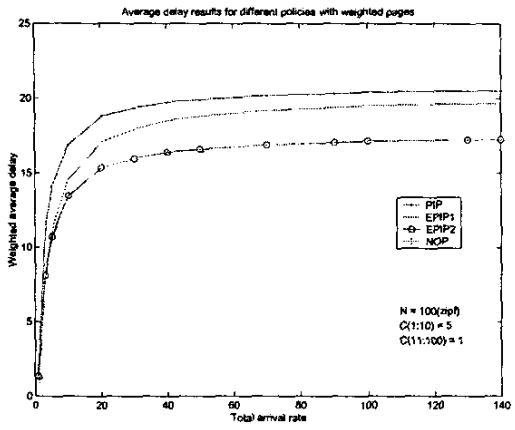

Fig. 6. Performance comparison of NOP and different versions of the PIP policy for the weighted average delay case.

NOP policies for other arrival distributions. To generate new distributions with maximum *contrast* to the convex-decreasing shape of the Zipf distribution, we chose two new distributions for the arrival rates covering the same range (1 to 100) but having linear-decreasing and concave-decreasing shapes. The resulting average waiting times under both policies were again almost identical for both of these new cases. It should be noted that the optimal index function is not unique and any monotonic increasing function of that function can be also used as an index function. Therefore, based on the above observation we conjecture that the PIP index is approximately a monotonic increasing function of the optimal index policy. In another set of experiments we compared the performances of PIP and NOP policies for the case where the pages have different weights. We showed in previous sections that the effect of weight C in the index function $\nu(s)$ is in the form of a simple multiplicative factor. PIP, in its original form, does not address the case with weights. Therefore, we tried to use the same analogy and extend its definition so that the weight coefficient appears in the index function as well. In the first extension, which we call EPIP1 for notational convenience, we define the index function as $\nu(x) = \frac{cx}{\sqrt{\lambda}}$ and in the second extension(EPIP2) we define it as $\nu(x) = \frac{\sqrt{c}x}{\sqrt{\lambda}}$. We performed the experiments on a system with 100 pages with Zipf distribution of the arrival rates and assigned a weight of 5 to the first 10 pages. The weights of the other pages were set to 1. Figure 6 shows the performances of all four policies under different arrival rates. As we can see, PIP by itself does not perform very well which is not unexpected. EPIP1, which uses the same multiplicative form as NOP to incorporate the effect of weights, also does not perform as good as NOP. However, EPIP2 have exactly the same performance as NOP and suggests that the effect of weight in the PIP index should be through a square root multiplicative factor.

VI. Conclusion and Discussion

In this report we derived an scheduling policy for the scheduling of broadcasts in a *pull* system. The policy defines an index function for each page in the system and at every decision instant the first K pages with the largest values of the index are broadcast. The performance of our policy is almost identical to the performance of the PIP policy however, since we have taken an analytical path for its derivation, it can be readily applied to cases with non-Poisson arrivals or when there are priority weights assigned to the pages. Other policies, due to their heuristic reasoning , do not address these general cases. Our approach shows that the index function scales linearly with the c coefficient. Using this result and through a number of experiments we also came up with a heuristic extension to the PIP to include the weighted case as well. Another advantage of our approach is the guidelines it provides to consider the scheduling problem for a system where the pages do not have fixed lengths. This case is particularly of interest for cache broadcasting in the Internet and the previous methods do not address this important case. We are currently working on this case and will publish the results in another report.

References

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

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

[3] K. Stathatos et. al., "Adaptive data broadcast in hybrid networks," *Proc. 23rd VLDB conf.*, Athens, Greece, 1997.

[4] et.al. H.D. Dykeman, "Scheduling algorithms for videotex systems under broadcast delivery," *IEEE Int. Conf. on Comm. ICC86*, Vol. 3,pp1847-51, 1986.

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

[6] 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.

[7] J.W. Wong M.H. Ammar, "The desgning of teletext broadcast cycles," *Perf. Eval. Rev.*, pp. Vol. 5, pp235–242, 1985.

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

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

[10] S. Hameed N. Vaidya, "Scheduling data broadcast in asymmetric communication environments," *Tech. report TR96-022, Dept. Computer Sci. Texas A and M Univ.*, 1996.

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

[12] M. J. Donahoo et. al., "Multiple-channel multicast scheduling for scalabel bulk-data transport," *INFOCOM'99*, pp847-855, 1999.

[13] Q. Hu et. al., "Dynamic data delivery in wireless communication environments," *Workshop on Mobile Data Access*, pp213-224, Singapore, 1998.

[14] M. Franklin D. Aksoy, "Scheduling for large-scale on-demand data broadcasting," *Proc. INFOCOM 98*, Vol. 2, pp651-9, 1998.

[15] M. Chaudhry and J. Templeton, *A First Course in Bulk Queues*, Wiley, New York., 1983.

[16] J. C. Gittins, *Multi-Armed Bandit Allocation Indices*, John Wiley & Sons, 1989.

[17] P. Whittle, "Multi-armed bandits and the gittins index," *J Roy. Statist. Soc. Ser. B.*, Vol. 42 pp143-149, 1980.

[18] P. Varaiya, J. Walrand, and C. Buyukkoc, "Extensions of the multi-armed bandit problem," *IEEE Transactions on Automatic Control AC-30*, pp426-439, 1985.

[19] J. C. Gittins, "Bandit processes and dynamic allocation indices," *J. Roy. Statist. Soc.*, Vol. 41, pp148-177, 1979.

[20] Jose Nino-Mora, "Restless bandits, partial conservation laws and indexability," *http://www.econ.upf.es/ ninomora/.*

[21] M. Putterman, *Markov Decision Processes : Discrete Stochastic Dynamic Programming*, Wiley, New York., 1994.

[22] J. Beasley, "Lagrangean relaxation," *Chapter 6: Lagrangean relaxation. In Colin R. Reeves, editor, Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publications*, 1993.

[23] M. Raissi-Dehkordi and J. S. Baras, "Broadcast scheduling in information delivery systems," *Technical Report, Institute for Systems Research*, University of Maryland at College Park, http://www.isr.umd.edu, 2002.