Broadcast Scheduling for Information Distribution *

Chi-Jiun Su [a,**] Leandros Tassiulas [b,***] Vassilis J. Tsotras [c,****]

[a] *Department of Electrical Engineering, Polytechnic University, Brooklyn NY 11201*
E-mail: cjsu@isr.umd.edu

[b] *Department of Electrical Engineering, University of Maryland, College Park MD 20742*
E-mail: leandros@isr.umd.edu

[c] *Department of Computer Science, University of California, Riverside, CA 92521*
E-mail: tsotras@cs.ucr.edu

Broadcast data delivery is encountered in many applications where there is a need to disseminate information to a large user community in a wireless asymmetric communication environment. In this paper, we consider the problem of scheduling the data broadcast such that average response time experienced by the users is low. In a push-based system, where the users cannot place requests directly to the server and the broadcast schedule should be determined based solely on the access probabilities, we formulate a deterministic dynamic optimization problem, the solution of which provides the optimal broadcast schedule. Properties of the optimal solution are obtained and then we propose a suboptimal dynamic policy which achieves average response time close to the lower bound. The policy has low complexity, it is adaptive to changing access statistics, and is easily generalizable to multiple broadcast channels. In a pull-based system where the users may place requests about information items directly to the server, the scheduling can be based on the number of pending requests for each item. Suboptimal policies with good performance are obtained in this case as well. Finally, it is demonstrated by a numerical study that as the request generation rate increases, the achievable performance of the pull- and push-based systems becomes almost identical.

Keywords: Broadcast Data Delivery, Information Distribution methods, Data Push/Pull, Multicast.

1. Introduction

Broadcast data delivery is rapidly becoming the method of choice for disseminating information to a massive user population in many new application areas where client-to-server communication is limited. This is due to communication asymmetry — physical asymmetry and/or information flow asymmetry — inherent in these applications. The main advantage of broadcast delivery is its scalability: it is independent of the number of users the system is serving. Some examples of the applications in which data broadcasting plays an important role are traffic information systems [18], information dispersal systems for volatile time-sensitive information such as stock prices and weather information [16], and news distribution systems [11]. In [12] and [13], data broadcasting is also considered as an efficient way, in terms of energy and bandwidth, for the distribution of information to a large number of users in a wireless communication environment.

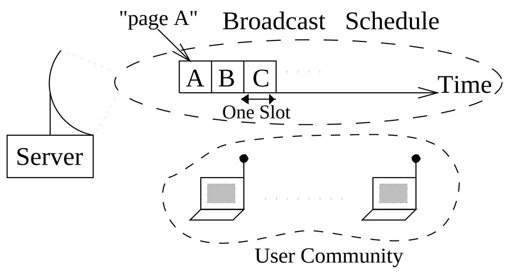

Figure 1. *A Broadcast Data Delivery System in a Wireless Communication Environment*

* Part of this paper was presented in INFOCOM'97, Kobe, Japan

** C.-J. Su is currently with Institute for Systems Research, University of Maryland, College Park on leave from Polytechnic University.

*** This research was supported in part by an NSF CAREER award NCR-9502614, by the AFOSR under grant 95-1-0061 and by an ONR YOUNG INVESTIGATOR award N000149710735.

**** The work of V.J. Tsotras was performed while the author was with Polytechnic University, Brooklyn, and it was partially supported under ARPA Contract No. DAAH01-94-C-R116, by NSF under grants No. IRI-9111271 and IRI-9509527, and by the New York State Science and Technology Foundation as part of its Center for Advanced Technology program.

In a broadcast data delivery system, depicted in figure 1, a server is continuously and repeatedly broadcasting data to a user community. There are two basic architectures for a broadcast delivery system: *push-based* broadcast delivery in which users cannot inform the server about what they actually need due to the lack of, or, limited uplink communication channel from the users to the server and *pull-based* broadcast delivery in which there is an uplink channel available through which a user can request from the server what it is waiting for.

Information broadcast by the server is organized into units called *pages*. Time on the broadcast channel is divided into slots of same size that is equal to the time to broadcast a page. When a user needs a certain page, it waits until the desired page appears on the broadcast and captures it for use. Hence, there is some latency from the time the need of a page arises until the time the page is actually broadcast by the server. This latency depends on the broadcast schedule. For a push-based system, due to the limitation imposed by the asymmetric communication channel, the server may know only the past access pattern of the users or an estimate of the user's access probability. The server relies on this information and broadcasts the pages according to a schedule that results in low latency for the user's requests. For the pull-based system, the server knows the exact number of pending requests for each page at each slot and can make use of the page request backlog information to decide which page to broadcast at each slot so as to minimize the response time experienced by a user.

Two major issues arise in data delivery systems: a) the organization of the data in a broadcast schedule so as to minimize the *average response time* ([10], [5], [6], [21], [1], [2], [8], [14], [19] and [9]) and b) the user's memory management in order to reduce the mismatch between the broadcast schedule and user's access pattern ([20], [22], [1], [3] and [4]). We addressed the latter problem in [20] where the optimal memory management policy was identified. Here, we concentrate on the first problem, i.e., how to design broadcast schedules in order to minimize the average response time of user's requests for both push-based and pull-based systems.

The problem of schedule design for broadcast information distribution systems has been studied in the past ([5], [6], [9], [10], and [21]). The motivation for that work was teletext systems. In [6] and [5], Ammar and Wong, using a stochastic Markov Decision Process (MDP) formulation, concluded that the optimal schedule for a push-based broadcast system will be periodic. They also proposed a method for designing periodic schedules with near optimal performance. In [9], the pull-based system was studied and several scheduling policies were evaluated.

In this paper, we formulate the scheduling problem in a push-based system as a deterministic MDP. Dynamic scheduling policies are considered where the scheduling decision at a slot is based on the elapsed time since the last transmission of each page. Properties of the optimal policy are identified. Furthermore, a class of policies are identified which have near optimal performance, of the same level or slightly better than the periodic scheduling policies proposed in [6]. Our policies have the advantage of being simple to implement in a real time fashion, adaptive to changes in the access statistics, and readily generalizable to systems with multiple parallel broadcast channels. In a pull-based system, the problem is formulated as a stochastic MDP. Properties of the optimal policy are identified and variations of the real-time scheduling schemes considered for the push-based system are evaluated and compared with previous results. Comparing the performance of push-based and pull-based systems, we observed that in certain cases and for sufficiently large request generation rates, the performance of the two systems is at about the same levels. That is, the availability of feedback channel for request placement capability does not improve the performance significantly at heavy load.

The paper is organized as follows. The problem is formulated in section 2. In section 3, the push-based system is studied. The pull-based system is investigated in section 4. Finally, generalization of our results for a multi-channel system is given in section 5.

2. The Broadcast Model

Slot n is the interval $[n, n+1)$. At each slot n, one page is broadcast in the channel and it is denoted by u_n, $u_n \in \{1, ..., M\}$ where M is the total number of possible pages. (The results easily generalize to the case of J parallel broadcast channels as it is discussed in section 5).

Requests for pages are generated by the users. A request for page i generated at time t will be satisfied at the next slot after t at which page i will be broadcast. Let $l_i(t)$ denote the number of slots from the end of slot n, where $t \in [n, n+1)$, until the end of the next slot after n at which the page is broadcast as shown in figure 2. Note that the latency of the request is equal to $l_i(t) + (n+1-t)$. Since the residual time $n+1-t$ from the generation of the request until the end of the slot is independent of the broadcast schedule, we will ignore it in the following and we will use $l_i(t)$ as the measure of the latency that will be experienced by a page i request generated at t.

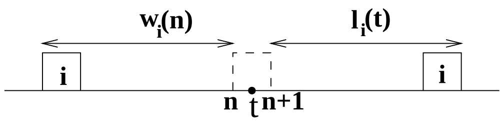

Figure 2. *Illustration of $l_i(t)$ and $w_i(n)$ for a sequence of page i broadcasts*

Let $d_i(k,t)$ be the delay, that has been experienced by a page i request generated at time k, $k \leq t$, until time t.

$$d_i(k,t) = \min\left(l_i(k), t - k'\right)$$

where $k \in [k'-1, k')$. Denote the sequence of times at which page i requests are generated as $t_n^i, n = 1, 2, ...$ for each page $i = 1, ..., M$. The aggregate delay, $D_i(t)$, of all page i requests up to time t, is

$$D_i(t) = \sum_{t_n^i \leq t} d_i(t_n^i, t).$$

Without loss of generality we assume in the rest of the paper that t is an integer. Let $X_i(n)$ be the total number of pending requests for page i at the beginning of slot n. The aggregate delay experienced by all page i requests up to time t is related to the page i request backlog as follows:

$$D_i(t) = \sum_{s=0}^{t-1} X_i(s) \tag{2.1}$$

The above formula is essentially Little's law for our system and its validity can be easily verified by figure 3 which shows a sample path of the evolution of page i request generation. The request generation instants correspond to the jumps of the curve which are of magnitude 1. The aggregate delay of page i requests up to time t equals the total shaded area under the curve in

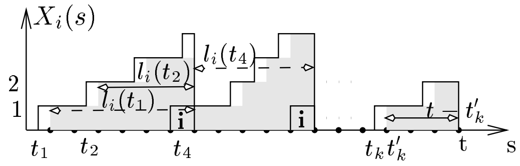

Figure 3. *The evolution of page i request backlog up to time t is depicted. At the end of each page i broadcast, all pending requests except those that are generated during the page i transmission are granted. The delay up to time t, $d_i(t_j, t)$, of page i request generated at time t_j is also depicted .*

the figure. Note that all the page i requests generated during a page i broadcast are assumed to have to wait until the next page i transmission.

Consider the aggregate stream of page requests generated by the whole user population. In the finite user population case, the rate of page request generation is affected by the number of users who are waiting for a page broadcast by the server. Since they will not generate a new page request while they are waiting, the rate of request generation will drop as the number of pending requests increases. If the user population though is large enough and an individual user request generation rate is appropriately normalized such that the aggregate rate is equal to λ, then we may assume that the aggregate page request generation rate remains constant and independent of the number of pending requests while the process of request generation is stationary.

A request is for page i with probability b_i, $i = 1, ..., M$, where $\sum_{i=1}^{M} b_i = 1$. Hence, requests for page i are generated according to a stationary process with rate $\lambda_i = b_i\ \lambda$. Let $A_i(n)$ be the total number of requests for page i generated during slot n. The request backlog for page i evolves as follows:

$$X_i(n+1) = \begin{cases} A_i(n) & \text{if } u_n = i \\ X_i(n) + A_i(n) & \text{otherwise} \end{cases} \tag{2.2}$$

The push-based and pull-based systems are considered separately next.

3. The Push-based Broadcast System

When the server is not aware about the user's requests, the broadcast schedule is designed based only on the distribution of page requests, that is, b_i, $i = 1, ..., M$. Designing the broadcast schedule to minimize delay is a static optimization problem that can be solved *off-line*. In [5], a schedule design method was proposed that results in schedules with good performance. Here, we formulate the schedule design as a deterministic dynamic optimization problem. The solution to the dynamic problem leads to computationally simple *on-line* scheduling policies.

Let $w_i(n)$ be the elapsed time from the beginning of the last transmission of page i before n until the beginning of slot n, as illustrated in figure 2. The evolution of $w_i(n)$ can be given as follows:

$$w_i(n+1) = \begin{cases} 1 & \text{if } u_n = i \\ w_i(n)+1 & \text{otherwise} \end{cases}$$

Assume that $w_i(0) = 1$ for $i = 1, ..., M$ without loss of generality. Hence, $w(n)$, $n = 0, 1, ...$ is a deterministic MDP controlled by u_n.

By taking expectations on both sides of equation (2.1), the expected aggregate delay of page i requests up to time t is

$$\overline{D_i}(t) = \sum_{s=1}^{t} \overline{X_i}(s).$$

where a variable with a bar on top represents the expected value of the variable.

The pending requests for page i, $X_i(s)$, are accumulated starting from the beginning of the last page i transmission before s. Since pages are generated by a stationary process with rate λ_i, we have $\overline{X_i}(s) = \lambda_i w_i(s)$. The evolution of the expected request backlog of page i is shown in figure 4.

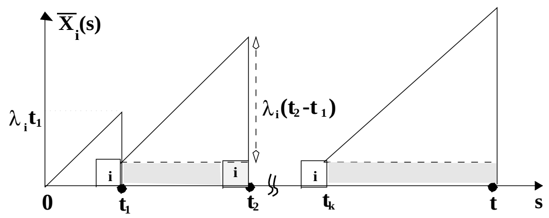

Figure 4. *The expected backlog is depicted as a function of time from time 0 to time t. At the end of each page i broadcast, there is some remaining page i request backlog (shaded portion under the curve) which accounts for the requests generated during the page i transmission.*

The expected aggregate delay can be written as

$$\overline{D_i}(t) = \sum_{n=1}^{t} \lambda_i w_i(n).$$

The long-term average delay is

$$D^u = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{n=1}^{T} \sum_{i=1}^{M} \lambda_i w_i(n) \tag{3.1}$$

where the superscript u signifies the dependence on the transmission schedule. The optimal transmission schedule is the one that minimizes the long-term average delay in (3.1).

From the MDP theory [7], it follows that the optimal schedule can be specified in terms of a scheduling policy, that is, a function $u : \mathcal{W} \to \mathcal{U}$ such that

$$u_n = u\,(w(n))$$

where $\mathcal{W} = N^M$ and $\mathcal{U} = \{1, ..., M\}$. Hence, characterizing the optimal schedule is equivalent to characterizing the function $u(\cdot)$. In the following, we show some properties of the optimal schedule.

3.1. Properties of the Optimal Policy

In order to study the optimization problem associated with the long run average cost (3.1), we need to consider first the optimization problem associated with the β-discounted cost. The β-discounted cost associated with a policy u is defined by

$$\begin{array}{l} V_u^\beta(w) \triangleq \sum_{t=0}^{\infty} \beta^t c\,(w(t), u(t))\,, \\ w(0) = w, \quad w \in \mathcal{W}, \quad 0 < \beta < 1. \end{array} \tag{3.2}$$

$c\,(w, u)$ is the cost incurred when the system is in state w and the action taken (the page broadcast by the server) is u, $u \in \mathcal{U}$ and is given by

$$c\,(w, u) \triangleq \sum_{i=1}^{M} 1\{i \neq u\}\; w_i \lambda_i$$

where $1\{.\}$ is an indicator function.

A scheduling policy, u^β, is said to be β-optimal if it minimizes (3.2), i.e., if for any other policy u

$$V_{u^\beta}^\beta(w) \;\leq\; V_u^\beta(w), \;\; w \in \mathcal{W}.$$

It is well known [7] that for the cost structure of our problem a stationary β-optimal policy exists. The β-optimal cost is by definition

$$V^\beta(w) = \inf_u V_u^\beta(w) \quad w \in \mathcal{W}$$

and satisfies the dynamic programming equation which, for our problem, is

$$V^\beta(w) =$$

$$\sum_{i=1}^{M} \lambda_i w_i + \min_{u \in \mathcal{U}} \left\{ -\lambda_u w_u + \beta V^\beta (w + \mathbf{1} - w_u \mathbf{e}_u) \right\}$$

where $\mathbf{1}$ is a vector with all entries equal to one and $\mathbf{e}_i$ is a vector with all its elements equal to zero except the ith element which is one. The β-optimal scheduling policy is

$$u^\beta(w) = \arg\min_{u \in \mathcal{U}} \left\{ -\lambda_u w_u + \beta V^\beta (w + \mathbf{1} - w_u \mathbf{e}_u) \right\}. \tag{3.3}$$

The first property of the β-optimal scheduling policy is that the priority of a page i to be scheduled for transmission at a slot n increases with $w_i(n)$. The threshold structure of the optimal policy, as it is expressed in the following lemma, reflects the above property.

Lemma 3.1. If $u^\beta\left(w^1\right) = j$, then for all w^2 such that $w_l^2 = w_l^1$, $l \neq j$, $l = 1, ..., M$, and $w_j^2 \geq w_j^1$, we have $u^\beta\left(w^2\right) = j$.

Proof. According to the assumption $u^\beta(w^1) = j$ and from (3.3),

$$\begin{aligned} & -\lambda_j w_j^1 + \beta V^\beta (w^1 + \mathbf{1} - w_j^1 \mathbf{e}_j) \\ & \leq -\lambda_k w_k^1 + \beta V^\beta (w^1 + \mathbf{1} - w_k^1 \mathbf{e}_k) \\ & \text{for} \quad k = 1, ..., M \end{aligned} \tag{3.4}$$

It is enough to show

$$\begin{aligned} & -\lambda_j w_j^2 + \beta V^\beta (w^2 + \mathbf{1} - w_j^2 \mathbf{e}_j) \\ & \leq -\lambda_k w_k^2 + \beta V^\beta (w^2 + \mathbf{1} - w_k^2 \mathbf{e}_k) \\ & \text{for} \quad k = 1, ..., M \end{aligned} \tag{3.5}$$

Since $w_l^1 = w_l^2$ for $l \neq j, l \in \{1, ..., M\}$ and $w_j^1 \leq w_j^2$, from (3.4) we have

$$\begin{aligned} & -\lambda_j w_j^2 + \beta V^\beta (w^2 + \mathbf{1} - w_j^2 \mathbf{e}_j) \\ & \leq -\lambda_k w_k^2 + \beta V^\beta (w^1 + \mathbf{1} - w_k^1 \mathbf{e}_k) \\ & \text{for} \quad k = 1, ..., M \end{aligned} \tag{3.6}$$

Note that if the same scheduling decisions are applied to two systems A and B with initial states $w^1 + \mathbf{1} - w_k^1 \mathbf{e}_k$ and $w^2 + \mathbf{1} - w_k^2 \mathbf{e}_k$ respectively, then the instantaneous cost in system A is less than or equal to that in system B. It can be easily concluded that

$$V^\beta (w^1 + \mathbf{1} - w_k^1 \mathbf{e}_k) \leq V^\beta (w^2 + \mathbf{1} - w_k^2 \mathbf{e}_k). \tag{3.7}$$

From (3.6) and (3.7), (3.5) follows. □

The next property is that, among the pages with the same request generation rates, priority is given to the page with the largest $w_i(n)$.

Lemma 3.2. . If $\lambda_i = \lambda_j$ and $w_i < w_j$, then $u^\beta(w) \neq i$.

Proof. By contradiction, assume that $u^\beta(w) = i$. Then,

$$\begin{aligned} & -\lambda_i w_i + \beta V^\beta (w + \mathbf{1} - w_i \mathbf{e}_i) \\ & \leq -\lambda_j w_j + \beta V^\beta (w + \mathbf{1} - w_j \mathbf{e}_j) \\ & \text{for } j = 1, ..., M \end{aligned} \tag{3.8}$$

Since $\lambda_i = \lambda_j$ and $w_i < w_j$, from (3.8)

$$V^\beta (w + \mathbf{1} - w_i \mathbf{e}_i) \ < \ V^\beta (w + \mathbf{1} - w_j \mathbf{e}_j).$$

If we apply the same scheduling decisions to two systems A and B with initial states $w + \mathbf{1} - w_j \mathbf{e}_j$ and $w + \mathbf{1} - w_i \mathbf{e}_i$ respectively except that page i is scheduled to transmit for system A whenever page j is scheduled for system B and vice versa, the instantaneous cost in system A is less than or equal to that in system B. Then it follows that

$$V^\beta (w + \mathbf{1} - w_j \mathbf{e}_j) \ \leq \ V^\beta (w + \mathbf{1} - w_i \mathbf{e}_i).$$

Therefore, it contradicts the assumption that $u^\beta(w) = i$ and the lemma is proved. □

Using standard techniques from the theory of Dynamic Programming we can extend the results of lemma 3.1 and 3.2 from the β-discounted cost to the long run average cost [17]. We state the results for the long run average cost optimal policies without a proof for the sake of brevity.

Theorem 3.3. If $u\left(w^1\right) = j$ minimizes the long run average cost (3.1), then for all w^2 such that $w_l^2 = w_l^1$, $l \neq j$, $l = 1, ..., M$, and $w_j^2 \geq w_j^1$, we have $u\left(w^2\right) = j$ as the optimal solution for (3.1).

Theorem 3.4. . If $\lambda_i = \lambda_j$ and $w_i < w_j$, then $u(w) = i$ does not optimize the long run average cost (3.1).

From theorem 3.4, it follows immediately that if all pages have the same request generation rate, the page i with the largest $w_i(n)$ will be transmitted at each slot n.

If there are only two pages, then the optimal policy can be completely characterized based on the threshold property expressed in theorem 3.3. Without loss of generality assume that $\lambda_2 \geq \lambda_1$.

Theorem 3.5. The optimal policy, when there are only two pages, is periodic with a period consisting of one transmission of page 1 followed by m ($m \geq 1$) transmissions of page 2 where

$$m = \max\left(1\ ,\ \arg\min_{l \in \mathcal{S}}\ D_o(l)\right).$$

$D_o(.)$ is the mean response time of such a policy and $\mathcal{S} = \left\{ \left\lfloor \sqrt{\frac{2\lambda_2}{\lambda_1}} - 1 \right\rfloor , \left\lceil \sqrt{\frac{2\lambda_2}{\lambda_1}} - 1 \right\rceil \right\}$.

Proof. Note that we can always improve a schedule with two consecutive transmission of page 1 by cancelling one of the transmissions and a schedule with different inter-appearance gaps between the transmissions of page 1 by selecting the gap with the lowest cost and constructing a schedule with identical gaps. Therefore, we only need to consider periodic policies with a period consisting of m consecutive transmissions of page 2 followed by a single transmission of page 1.

Mean response time of the periodic schedule with period $m+1$ is

$$D_{avg} = \frac{1}{(m+1)} \left\{ \frac{1}{2}\lambda_1 (m+1)^2 \ +\ (m-1)\ \frac{1}{2}\lambda_2\ +\ \frac{1}{2}\lambda_2 2^2 \right\}$$

Since D_{avg} is a convex function of m,

$$\arg\min_m D_{avg} = \sqrt{\frac{2\lambda_2}{\lambda_1}} - 1$$

and the theorem is proved. □

Specifying the exact form of the optimal scheduling policy appears to be an intractable problem in general. In the following, we specify a class of scheduling policies that incorporate some of the characteristics of the optimal policy shown above and they turn out to achieve average response time close to the lower bound.

3.2. Near Optimal Real Time Scheduling

There are two quantities related to each page i that affect the scheduling decision at each slot n. The elapsed time $w_i(n)$ since the last transmission of page i and the rate λ_i of request generation for page i. The likelihood of page i being transmitted at n increases with λ_i and $w_i(n)$. We consider the policies where the broadcast scheduling is determined based on priority indices of the pages. The index of page i is the product $\lambda_i^\gamma\ w_i(n)$ where γ is an exponent that reflects the relative importance of λ_i versus $w_i(n)$ in determining the priority. The page scheduled to be broadcast at slot n is

$$u_n = arg \max_{i \in \{1,\ldots,M\}} \ \lambda_i^\gamma\ w_i(n) \tag{3.9}$$

In the rest we refer to the above class of policies as the *priority index policies*. Note that when all the pages have identical request generation rates, the priority index policies for all γ's generate uniform periodic schedules which are optimal in this case.

Certain policies are worth distinguishing among the priority index policies. For $\gamma = 0$, the dependence of the scheduling decision on the request generation rate vanishes and the resulting schedule is periodic with each page being transmitted once in each period. For $\gamma = 1$, the index $\lambda_i w_i(n)$ of page i is equal to the expected backlog of page i, $\overline{X_i}(n)$, and the policy schedules the page with the largest backlog at each slot n. For $\gamma = 0.5$, the index of page i is $\lambda^{0.5} w_i(n) = \sqrt{\lambda_i w_i^2(n)}$. Note that $\frac{1}{2}\lambda_i w_i^2(n)$ is the aggregate expected delay experienced by page i requests since the last transmission of page i before slot n. Hence, for $\gamma = 0.5$, the page with the largest *Mean Aggregate Delay* (*MAD*) is selected for transmission.

We performed an extensive numerical study of the performance of the system under the priority index policies for various values of γ. It turns out that the *MAD* policy ($\gamma = 0.5$) has the best performance in most cases. Furthermore, the performance of MAD is very close to the lower bound on the mean response time, that was given in [5].

3.2.1. Performance Comparisons Among Priority Index Policies

Comparisons are made for $M = 100$ to $M = 1000$ and $\gamma = 0$ to 1.0 for the following two cases. In the first case, user access probabilities are assumed to follow the zipf distribution version I [23], that is, $b_i = \frac{c}{i}$, $i = 1, ..., M$ where c is a normalizing constant given by $c = (\sum_{j=1}^{M} 1/j)^{-1}$. In the second case, we assume page access probabilities follow the zipf distribution version II [15] where $b_i = \frac{i^\theta - (i-1)^\theta}{M^\theta}$ where $i = 1, ..., M$. As θ decreases, the access pattern becomes increasingly skewed. The value of θ used in this experiment is $\log(0.8)/\log(0.2)$. Zipf distribution is typically used to model non-uniform access patterns. The mean response time for the heuristic policies is also compared to the lower bound for a periodic broadcast schedule which is obtained in [5] and is given by

$$\frac{1}{2}\left(\sum_{i=1}^{M}\sqrt{b_i}\right)^2.$$

Table 1
Mean Response Time in slots *for different values of γ using zipf distribution I (L. B. denotes Lower Bound)*

M	1	0.75	0.6	0.5	0.25	0	L. B.
100	48.49	36.61	33.82	33.36	37.60	50.0	33.31
200	97.56	68.92	62.52	61.41	70.42	100.0	61.36
300	145.21	99.75	89.58	87.81	101.72	150.0	87.77
400	193.69	129.72	115.67	113.22	131.65	200.0	113.18
500	244.29	159.06	141.07	137.93	161.15	250.0	137.90
600	295.68	187.86	165.93	162.11	190.65	300.0	162.08
700	343.00	216.37	190.37	185.86	218.91	350.0	185.83
800	389.06	244.60	214.45	209.25	246.71	400.0	209.21
900	437.05	272.26	238.23	232.34	274.10	450.0	232.29
1000	486.86	299.88	261.74	255.15	301.38	500.0	255.13

Table 2
Mean Response Time in slots *for different values of γ using zipf distribution II*

M	1	0.75	0.6	0.5	0.25	0	L. B.
100	46.63	27.02	23.63	23.14	28.31	50.0	23.08
200	94.62	53.08	46.19	45.13	55.28	100.0	45.03
300	136.23	78.94	68.57	66.95	81.55	150.0	66.86
400	175.98	105.59	91.01	88.69	108.02	200.0	88.58
500	216.31	130.74	113.23	110.35	134.00	250.0	110.25
600	262.66	156.73	135.43	132.06	160.41	300.0	131.94
700	310.82	183.71	157.65	153.66	186.10	350.0	153.57
800	359.62	207.51	179.84	175.22	211.90	400.0	175.18
900	409.25	233.77	202.00	196.80	237.27	450.0	196.77
1000	459.46	260.59	224.17	218.38	262.52	500.0	218.32

According to the results from Table 1 and 2, the MAD policy ($\gamma = 0.5$) yields the best performance which is also close to the lower bound for both distributions. MAD also gave the best mean response time among the priority index polices for various distributions we have tried (not mentioned here) and its performance is consistently close to the lower bound.

Although the algorithm proposed in [5] also yields a mean response time close to the lower bound, the MAD policy has a number of advantages over other existing methods for designing broadcast schedules. First, it automatically generates broadcast schedules without requiring to select the three basic parameters of a periodic schedule — period length, appearance frequencies and inter-appearance gaps of each page in one period. Second, theMAD policy does not need to perform any precomputation before the broadcast. It selects the page to transmit at each slot during the broadcast according to the given user's access probabilities. Thus, the policy can adapt the schedules as the user access pattern changes. Third, as it is shown in section 5, it can be generalized easily for multi-channel systems.

Moreover, the MAD policy is easy to implement; it only needs to keep M values of $w_i(n)$ at each slot n in addition to the access probabilities and the only operations required to perform at each slot n are to update the values of $w_i(n)$ and to determine the page with the largest expected aggregate delay of the current request backlog. Therefore, both the computational complexity and the storage requirement of the MAD policy is just $O(M)$. On the other hand, the approach in [5] is an *off-line* algorithm which has to construct the whole schedule and store it before the broadcast. This may require a considerable storage when the period of a schedule is large, which is usually the case when there are a large number of pages and page access probabilities are non-uniform. Furthermore, it is easy to show that the MAD policy generates schedules that are periodic.

4. The Pull-based Broadcast System

When there is an uplink channel available for the users to submit page requests, the server knows the exact number of pending requests for each page at each slot and it can make the scheduling decision based on that information. The request backlog vector $X(n)$ evolves according to equation (2.2) as well. The scheduling decision u_n at slot n may depend on the backlog evolution up to slot n. The optimal scheduling policy is the one that minimizes

$$\lim_{T\to\infty}\sup\frac{1}{T}\sum_{n=0}^{T-1}\sum_{i=1}^{M}\overline{X_i}(n). \tag{4.1}$$

The MDP theory suggests that the optimal scheduling policy can be specified in terms of a function $u : \mathcal{X} \to \{0, 1, ..., M\}$ such that

$$u_n = u(X(n))$$

where $\mathcal{X} = Z_+^M$.

The transition probability of the state process is given by

$$P_{XY}(u) = \begin{cases} p_{A'} \text{ if } Y = X + A' - X_u \mathbf{e}_u \\ 0 \quad \text{Otherwise} \end{cases}$$

where $p_{A'}$ is the probability that each element A'_i, $i = 1, ..., M$, of the vector A' is equal to the number of page i requests generated in one slot. The cost incurred when the system is in state X and the action taken is u, $u \in \mathcal{U}$, is

$$c(X, u) \triangleq \sum_{i=1}^{M} 1\{i \neq u\} \; X_i.$$

The β-discounted cost associated with a policy $u \in U$ can be defined in the similar way as in section 3 and the β-optimal cost associated with scheduling policies satisfies the following dynamic programming equation:

$$V^\beta(X) = \min_{u \in \mathcal{U}} \left\{ c(X, u) + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X + A' - X_u \mathbf{e}_u) \right\}.$$

The β-optimal scheduling policy is given by

$$u^\beta = \arg\min_{u \in \mathcal{U}} \left\{ c(X, u) + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X + A' - X_u \mathbf{e}_u) \right\} \tag{4.2}$$

The β-optimal scheduling policy exhibits properties similar to those of the push-based system. The first property is that the priority of a page i to be scheduled for transmission at a slot n increases with $X_i(n)$ and it is expressed in the following lemma.

Lemma 4.1. . If $u^\beta(X^1) = j$, then for all X^2 such that $X_l^2 = X_l^1$, $l \neq j$, $l = 1, ..., M$, and $X_j^2 \geq X_j^1$, we have $u^\beta(X^2) = j$.

Proof. Since we have assumed that $u^\beta(X^1) = j$ and from (4.2),

$$\begin{aligned} &-X_j^1 + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X^1 + A' - X_j^1 \mathbf{e}_j) \\ &\leq -X_k^1 + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X^1 + A' - X_k^1 \mathbf{e}_k) \\ &\text{for} \quad k = 1, ..., M \end{aligned} \tag{4.3}$$

To prove the lemma, we only need to show that

$$\begin{aligned} &-X_j^2 + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X^2 + A' - X_j^2 \mathbf{e}_j) \\ &\leq -X_k^2 + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X^2 + A' - X_k^2 \mathbf{e}_k) \\ &\text{for} \quad k = 1, ..., M \end{aligned} \tag{4.4}$$

Since $X_l^2 = X_l^1$ for $l \neq j$, $l = 1, ..., M$, and $X_j^2 \geq X_j^1$, from (4.3) we have

$$\begin{aligned} &-X_j^2 + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X^2 + A' - X_j^2 \mathbf{e}_j) \\ &\leq -X_k^2 + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X^1 + A' - X_k^1 \mathbf{e}_k) \\ &\text{for} \quad k = 1, ..., M \end{aligned} \tag{4.5}$$

If the same scheduling decisions are applied to two systems A and B with initial states $X^1 + A' - X_k^1 \mathbf{e}_k$ and $X^2 + A' - X_k^2 \mathbf{e}_k$ respectively and the request generation process is identical for both systems, then since $X^1 + A' - X_k^1 \mathbf{e}_k \leq X^2 + A' - X_k^2 \mathbf{e}_k$ in element-wise sense, it follows that the β-discounted cost in system A is less than or equal to that in system B,

$$\begin{aligned} &V^\beta(X^1 + A' - X_k^1 \mathbf{e}_k) \\ &\quad \leq V^\beta(X^2 + A' - X_k^2 \mathbf{e}_k) \quad \forall A' \in Z_+^M. \end{aligned} \tag{4.6}$$

From (4.5) and (4.6), (4.4) follows. □

Another property of the β-optimal scheduling policy is that among the pages with the same request generation rate, priority for transmission is given to the page with the largest backlog.

Lemma 4.2. . If $\lambda_i = \lambda_j$ and $X_i < X_j$, then $u^\beta(X) \neq i$.

Proof. We will give the proof by contradiction. Assume $u^\beta = i$. Then,

$$\begin{aligned} &-X_i + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X + A' - X_i \mathbf{e}_i) \\ &\leq -X_j + \beta \sum_{A' \in Z_+^M} p_{A'} V^\beta(X + A' - X_j \mathbf{e}_j) \\ &\text{for} \quad j = 1, ..., M \end{aligned} \tag{4.7}$$

Since $Xi < X_j$, from (4.7)

$$\sum_{A' \in Z_+^M} p_{A'} V^\beta(X + A' - X_i \mathbf{e}_i) < \sum_{A' \in Z_+^M} p_{A'} V^\beta(X + A' - X_j \mathbf{e}_j)$$

If we apply the same scheduling decisions to two systems A and B with initial states $X + A' - X_j\mathbf{e}_j$ and $X + A' - X_i\mathbf{e}_i$ respectively except that page i is scheduled to transmit for system A whenever page j is scheduled for system B and vice versa and both systems have the identical request generation process, then the β-discounted cost in system A is less than or equal to that in system B,

$$V^\beta(X + A' - X_j\mathbf{e}_j) \ \le \ V^\beta(X + A' - X_i\mathbf{e}_i) \quad \forall A' \in Z_+^M.$$

Therefore, it contradicts the assumption that $u^\beta(w) = i$ and the lemma is proved. □

The following theorems follow from lemmas 4.1 and 4.2 using standard methods to relate the β-discounted and the long run average cost problems in [17].

Theorem 4.3. If $u\left(X^1\right) = j$ minimizes the long run average cost (4.1), then for all X^2 such that $X_l^2 = X_l^1$, $l \neq j$, $l = 1, ..., M$, and $X_j^2 \geq X_j^1$, we have $u\left(X^2\right) = j$ as the optimal solution for (4.1).

Theorem 4.4. . If $\lambda_i = \lambda_j$ and $X_i < X_j$, then $u(X) = i$ does not minimize the long run average cost (4.1).

An immediate consequence of theorem 4.4 is that if all pages have the same request generation rate, the optimal policy is to broadcast the page with the largest backlog at each slot.

For arbitrary request generation rates, the optimal policy appears to resist a simple characterization. We studied a class of heuristic scheduling policies which are of the same flavor as the priority index policies employed in the push-based system.

They are described as follows:

$$u_n = \arg \max_{i \in \{1,\ldots,M\}} \ \lambda_i^{-\gamma} \ X_i(n)$$

Similar to the push-based system, when all the request generation rates are equal, the priority index scheduling policies also produce the optimal schedule for the pull-based system.

A number of heuristic scheduling policies for the push-based system were proposed in [9]. Two of them are the *Most Request First* (*MRFL*) policy, which selects the page with the largest number of pending requests and breaks ties in favor of the lowest probability page, and the *Longest Wait First* (*LWF*) policy, which selects the page for which the total waiting time of pending requests is the largest. The *MRFL* policy corresponds to the priority index policy with $\gamma = 0$. Since, according to the simulation results in [9], the *LWF* policy yields significantly better response time characteristics than other heuristic policies, we compare the priority index policies to the *LWF* policy by simulation. The results for 1000 pages with zipf I and zipf II distribution are shown in figure 5 and 6 respectively.

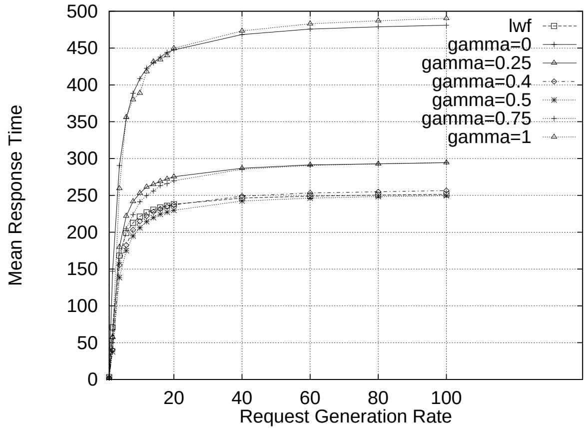

Figure 5. *Mean Response Time (in slots) vs. Aggregate Request Generation Rate (requests per slot) for different values of γ using zipf distribution I for 1000 pages*

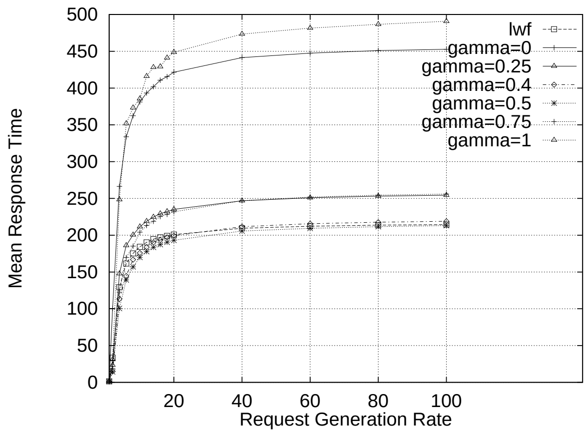

Figure 6. *Mean Response Time (in slots) vs. Aggregate Request Generation Rate (requests per slot) for different values of γ using zipf distribution II for 1000 pages*

For light load, the mean response time is insensitive to the particular scheduling algorithm employed. As the request generation rates increases, the policy with $\gamma = 0.5$ exhibits the best mean response time (even slightly better than the *LWF* policy) for all aggregate request generation rates. The policy with $\gamma = 0.4$ performs close to the *LWF* policy and the policy with $\gamma = 1.0$ gives the worst performance. Note that the policy with $\gamma = 0.5$ is easier to implement than the *LWF* policy since, at each slot, it only needs to keep track of the request backlog for each page while the *LWF* policy has to compute the total waiting time up to the current slot for each page.

4.1. Performance Limits of a Pull-Based Broadcast System

A pull-based system requires the availability of an uplink channel and has the undesirable property that the uplink channel may become overloaded under heavy aggregate request generation rate. Our simulation results show that the mean response time of a pull-based system approaches that of a push-based system as the aggregate request generation rate increases.

Figure 7 shows the simulation results for the case of equal request generation rates (λ_i is the same for all pages). As the aggregate request generation rate increases beyond 20, the mean response time of the pull-based system approaches half of the total number of pages which happens to be the mean response time of the optimal schedule for the push-based system. The intuitive explanation is as follows. For the case with the same generation rates for all pages, the optimal schedule for the push-based system is to broadcast the page with the largest $w_i(n)$ at each slot n while that for the pull-based system transmits the page with the largest number of pending requests $X_i(n)$ at each slot n. The expected value of page i request backlog increases linearly with both $w_i(n)$ and the page i request generation rate λ_i. Therefore, as the aggregate request generation rate increases, the probability that the backlog of page i in the pull-based system corresponding to the page with the largest $w_i(n)$ in the push-based system is the largest among the backlogs of all other pages in the pull-based system increases towards one as well. Hence, the probability that the page transmitted by both the push-based system and the pull-based system is the same, approaches one as the aggregate generation rate increases.

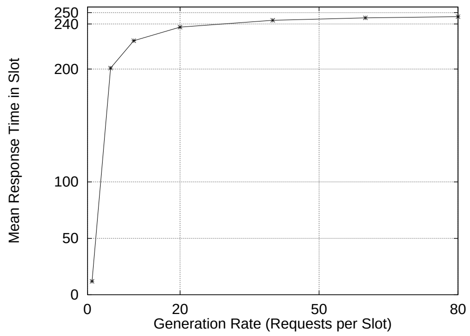

Figure 7. *Mean Response Time (in slots) vs. Aggregate Request Generation Rate (in request per slot) for 500 pages with equal generation rates*

For the case with unequal request generation rates, consider the policy with $\gamma = 0.5$ as an example for both pull-based and push-based systems. From figures 5 and 6, the mean response time of the policy with $\gamma = 0.5$ for 1000 pages with zipf distribution I and II for the pull-based system are 249.59 and 213.09 slots respectively at aggregate request generation rate 100 requests per slot while the mean response times for the push-based system for zipf distribution I and II are 255.12 and 218.36 slots respectively from tables 1 and 2. Hence as the aggregate request generation rate increases, the mean response time of a pull-based system is indeed approaching that of a push-based system for both zipf distributions I and II.

5. Application to a System with Multiple Broadcast Channels

Sometimes, since the available bandwidth for the wireless broadcast channel is considerably large, the channel has to be divided into a number of subchannels with smaller bandwidth due to implementation constraints. Therefore, there are more than one broadcast channels available and a number of pages equal to the number of channels is broadcast at each slot. Assume that all the users have the ability to tune in to any of the broadcast channels and retrieve the corresponding page. By using a small fraction of the bandwidth, the server may inform the users about which pages are being broadcast at which channels at each slot so that a user will know which channel he should tune in at each slot. A system with J broadcast channels is depicted in figure 8.

The problem of designing broadcast schedules for a push-based system in this case is different from the sin-

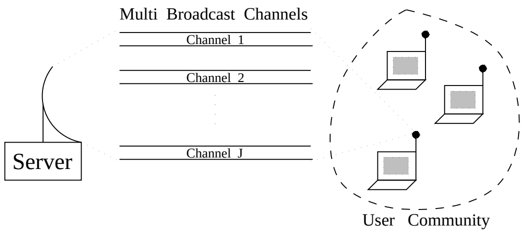

Figure 8. *A System with Multiple Broadcast Channels*

gle channel case. At each slot n, the server has to select J pages to broadcast to the users. The request backlog of any of the J pages broadcast during slot n vanishes at the end of slot n.

The existing algorithms for designing broadcast schedules for a push-based system in literature are only intended for a single-channel system and they cannot be easily extended for the multichannel case. The *MAD* policy, however, can be readily generalized for the multichannel case as follows.

At slot n select the J pages, for which the quantity $\lambda_j^{0.5} w_j(n)$ is largest and broadcast them. In the same way, the *MAD* policy can be applied to a pull-based system with multiple channels.

The lower bound to the mean response time for the single channel system can be readily generalized for a system with J broadcast channels to be

$$\frac{1}{2J}\left(\sum_{i=1}^{M}\sqrt{b_i}\right)^2$$

The mean response time of the schedules produced by *MAD* policy is compared to the lower bound for $M = 60$ to $M = 100$. For each M, we consider the number of channels to be approximately 5% and 10% of the total number of pages. The results are given in Table 3 and 4. The examples show that, if the number of broadcast channels is increased twofold, the mean response time decreases nearly by half. Note that, ideally, we would like to get the mean response time reduced exactly by half when the number of channels is doubled. In all the cases we consider, the schedules produced by the *MAD* policy incur mean response time close to the lower bound.

Table 3
Mean Response Time in slots *for zipf distribution I*

M	No. of Channels	Lower Bound	MAD
60	3	7.08	7.10
60	6	3.54	3.57
70	3	8.11	8.13
70	6	4.05	4.07
80	4	6.84	6.86
80	8	3.42	3.44
90	4	7.59	7.61
90	8	3.79	3.83
100	5	6.66	6.68
100	10	3.33	3.35

Table 4
Mean Response Time in slots *for zipf distribution II*

M	No. of Channels	Lower Bound	MAD
60	3	4.73	4.79
60	6	2.37	2.47
70	3	5.47	5.50
70	6	2.74	2.82
80	4	4.66	4.70
80	8	2.33	2.37
90	4	5.21	5.28
90	8	2.61	2.69
100	5	4.61	4.62
100	10	2.30	2.31

6. Conclusion

We considered the problem of scheduling data broadcasts such that the average response time experienced by the users is minimized. In a push-based system the problem was formulated as a deterministic MDP and properties of the optimal solution were obtained. A class of policies (the priority index policies) were examined and a suboptimal dynamic policy (*MAD*) that achieved average response time close to the lower bound was identified. Our policy has low implementation complexity, it is adaptive to changing access statistics and can be easily generalizable to multiple broadcast channels. Suboptimal policies with good performance were also obtained for a pull-based system. Interestingly enough, the numerical results showed that as the request rate increases the achievable performance of the push- and pull-based systems becomes almost identical; we plan to investigate this further.

Acknowledgments

The authors would like to thank the reviewers for their thoughtful comments that helped to improve the presentation of this paper.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. "Broadcast Disks: Data Management for Asymmetric Communication Environments". Technical Report CS-94-43, Dept. of Comp. Science, Brown University, October 1994.

[2] S. Acharya, M. Franklin, and S. Zdonik. "Dissemination-based Data Delivery Using Broadcast Disks". *IEEE Personal Communications*, 2(6):50–60, December 1995.

[3] S. Acharya, M. Franklin, and S. Zdonik. "Prefetching from a Broadcast Disk". In *Proc. 12th Int'l. Conf. Data Eng.*, New Orleans, LA, February 1996.

[4] M. H. Ammar. "Response Time in a Teletext System: an Individual User's Perspective". *IEEE Transaction on Communication*, COM-35(11):1159–1170, November 1987.

[5] M. H. Ammar and J. W. Wong. "The Design of Teletext Broadcast Cycles". *Performance Evaluation*, 5(4):235–242, December 1985.

[6] M. H. Ammar and J. W. Wong. "On the Optimality of Cyclic Transmission in Teletext Systems". *IEEE Transaction on Communication*, COM-35(1):68–73, January 1987.

[7] Dimitri P. Bertsekas. *"Dynamic Programming: Deterministic and Stochastic Models"*. Prentice-Hall, Inc., Englewood Cliffs, N.J.07632, 1987.

[8] T. Chiueh. "Scheduling for Broadcast-based File Systems". *Proc. of the Mobidata Workshop*, November 1994. Rutgers University, NJ.

[9] H. D. Dykeman, M. H. Ammar, and J. W. Wong. "Scheduling Algorithms for Videotex System under Broadcast Delivery". *Proceedings of ICC' 86*, pages 1847–1851, 1986.

[10] J. Gecsei. *"The Architecture of Videotex Systems"*. Prentice-Hall, Inc., Englewood Cliffs, N.J.07632, 1983.

[11] D. K. Gifford. "Polychannel Systems for Mass Digital Communication". *Communications of the ACM*, 33(2):141–151, February 1990.

[12] T. Imielinski and B. Badrinath. "Mobile Wireless Computing: Solutions and Challenges in Data Management". Tech. rep., Dept. of Compt. Sci., Rutgers University, NJ, 1992.

[13] T. Imielinski, S. Viswanathan, and B. Badrinath. "Energy Efficient Indexing on Air". *Proc. of ACM SIGMOD Conf.*, pages 25–36, May 1994.

[14] R. Jain and J. Werth. "Airdisks and AirRAID: Modelling and Scheduling Periodic Wireless Data Broadcast". Dimacs technical report 95-11, Computer Science Dept., Rutger University, May 1995.

[15] Donald E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, Reading, Massachusetts, second edition, 1981.

[16] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen. "The Information Bus-An Architecture for Extensible Distributed Systems ". *Proc. 14th SOSP*, December 1993.

[17] Sheldon M. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, New York, 1983.

[18] S. Shekhar and D. Liu. "Genesis and Advanced Traveler Information Systems ATIS: Killer Applications for Mobile Computing ". *MOBIDATA Workshop*, 1994.

[19] C.-J. Su, L. Tassiulas, and V. Tsotras. "A New Method to Design Broadcast Schedules in a Wireless Communication Environment". Technical report, Institute For Systems Research, University of Maryland, College Park, 1996.

[20] L. Tassiulas and C. J. Su. "Optimal Memory Management Strategies for a Mobile User in a Broadcast Data Delivery System". *IEEE JSAC Special Issue on Networking and Performance Issues of Personal Mobile Communications*, 15(7):1226–1238, September 1997.

[21] J. W. Wong. "Broadcast Delivery". *Proceedings of the IEEE*, 76(12):1566–1577, December 1988.

[22] S. Zdonik, S. Acharya, R. Alonso, and M. Franklin. "Are 'Disks in the Air' Just Pie in the Sky?". *IEEE Workshop on Mobile Computing Systems and Applications*, December 1994.

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

Biography of Chi-Jiun Su

Chi-Jiun Su received the B.S. degree in communication engineering from the National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1993, and the M.S. and Ph.D. degrees in electrical engineering from Polytechnic University, Brooklyn, NY, in 1996 and 1998 respectively.

He was awarded Special Polytechnic Graduate Fellowship in 1993 and was a research assistant at the Center for Advanced Technology in Telecommunications (CATT) at Polytechnic University. He is currently visiting the University of Maryland, College Park as a Faculty Research Assistant at the Institute for Systems Research. His main research interests include large scale information dissemination systems, mobile computing, design and management of wireless communication and high speed networks. Dr. Su is a student member of the IEEE and a member of Eta Kappa Nu.

Biography of Leandros Tassiulas

Leandros Tassiulas was born in 1965, in Katerini, Greece. He obtained the Diploma in Electrical Engineering from the Aristotelian University of Thessaloniki, Thessaloniki, Greece in 1987, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Maryland, College Park in 1989 and 1991 respectively. From 1991 to 1995 he was an Assistant Professor in the Department of Electrical Engineering, Polytechnic University, Brooklyn, NY. In 1995 he joined the Department of Electrical Engineering, University of Maryland, College Park, where he is now an Associate Professor . He holds a joint appointment with the Institute for Systems Research and is a member of the Center for Satellite and Hybrid Communication Networks, established by NASA. His research interests are in the field of computer and communication networks with emphasis on wireless communications and high-speed network architectures and management, in control and optimization of stochastic systems and in parallel and distributed processing.

Dr. Tassiulas received a National Science Foundation (NSF) Research Initiation Award in 1992 an NSF Faculty Early Career Development Award in 1995 and an Office of Naval Research Young Investigator Award in 1997. He coauthored a paper that got the INFOCOM '94 best paper award. Dr. Tassiulas is a member of the IEEE.

Biography of Vassilis Tsotras

Vassilis Tsotras received the B.Sc. degree in Electrical Engineering (1985) from the National Technical University of Athens, Greece and the M.Sc., M.Phil., and Ph.D. degrees in Electrical Eng. from Columbia University, in 1986, 1988, and 1990 respectively. In September 1990 he joined Polytechnic University, Brooklyn, as an Assistant Professor, and in 1996 he became a (tenured) Associate Professor of Computer and Information Science. During summer of 1997 he was on a sabbatical visit at the Department of Computer Science of the University of California, Los Angeles. He is currently with the Department of Computer Science at the University of California, Riverside. His research interests include temporal databases, access methods, data dissemination and parallel database systems. Dr. Tsotras received a National Science Foundation (NSF) Research Initiation Award in 1991.