Using Learning Automata for Adaptive Push-Based Data Broadcasting in Asymmetric Wireless Environments

Petros Nicopolitidis, Georgios I. Papadimitriou, *Senior Member, IEEE*, and Andreas S. Pomportsis

***Abstract*—Push systems are not suitable for applications with *a priori* unknown, dynamic client demands. This paper proposes an adaptive push-based system. It suggests the use of a learning automaton at the broadcast server to provide adaptivity to an existing push system while maintaining its computational complexity. Using simple feedback from the clients, the automaton continuously adapts to the client population demands so as to reflect the overall popularity of each data item. Simulation results are presented that reveal the superior performance of the proposed approach in environments with *a priori* unknown, dynamic client demands.**

***Index Terms*—Adaptive data broadcasting, asymmetric wireless environments, learning automata.**

I. Introduction

DATA broadcasting has emerged as an efficient way of disseminating information over asymmetric wireless environments [1], where client needs for data items are usually overlapping. In such cases, broadcasting is an efficient solution, since the broadcast of a single data item is likely to satisfy a (possibly large) number of clients.

The three approaches for designing broadcast schedules are pull, push, and hybrid. In pull systems (e.g., [2]), mobile clients make requests via the uplink channel. The server uses these requests to estimate the demand probability per data item and schedules its broadcasts accordingly. This approach has the advantage of adaptivity, since the server possesses knowledge regarding client demands. However, it is not scalable, since for large client populations, client requests will either collide with each other or saturate the server. In push systems, there is no interaction between the server and the mobile clients. The server is assumed to have an *a priori* estimate of the demand per information item and broadcasts items according to this estimate. Thus, push systems are not adaptive to dynamic[1] demands. However, they provide high scalability and client hardware simplicity since the client does not need to include packet transmission capability. Hybrid systems divide the available downlink bandwidth into two modes: the periodic broadcast mode, in which the server pushes data periodically to the clients; and the on-demand mode, which is used to broadcast data explicitly requested by mobile clients. To our knowledge, the only hybrid system that achieves adaptivity is proposed in [3] and its derivation in [4]. However, hybrid systems have to strike a careful balance between push and pull modes and impose the needs for client transmission capability and existence of a backchannel wide enough to carry client requests.

Until now, research on push-based systems assumed *a priori* knowledge of client demands. However, in today's information retrieval applications, overall client demands are likely to be unknown and change with time. Such an example is the case of flight information dissemination in an airport [5]. In such a scenario, users coming to the airport will want information regarding their flight (e.g., exact hour of departure, possible delays, etc.). A broadcast server should deliver data according to overall client demands. For a specific flight, the demand is likely to be in its peak a couple of hours before the flight's departure. For example, if our flight departs at 6 PM, early in the day the demand will be very small, as few passengers are likely to come to the airport five or six hours before their flight. At this time of the day, the server should increase the frequency of data items that concern other flights that leave in the near future. As the time for the departure of our flight approaches, the demand for information regarding it will grow due to the increasing number of waiting passengers. Eventually, a few minutes after the departure of the flight, it will drop again. It can easily be seen that in such an environment, overall client demands are neither *a priori* known nor static.

This paper proposes an adaptive push-based system. It uses a learning automaton at the broadcast server to provide adaptivity to a nonadaptive (static) push approach [6] while maintaining its computational complexity. Through a simple feedback from the clients, the automaton continuously adapts to the overall client population demands in order to reflect the overall popularity of each data item.

The remainder of this paper is organized as follows. Section II overviews the static approach to which our method provides adaptivity. Section III briefly introduces learning automata. Next, it presents our approach for using a learning automaton as the core of the server adaptation mechanism. Section IV presents simulation results that reveal the superiority of the proposed approach in environments with dynamic client demands. Section V concludes this paper.

Manuscript received January 15, 2001; revised June 28, 2001, January 11, 2002, and April 15, 2002.

The authors are with the Department of Informatics, Aristotle University of Thessaloniki, 54006 Thessaloniki, Greece (e-mail: gp@csd.auth.gr).

Digital Object Identifier 10.1109/TVT.2002.802978

[1]Hereinafter, we use the term *dynamic* to characterize changing client demand patterns with the nature of occurrence of these changes' being unknown to the broadcast server.

II. Related Research

To optimize performance, broadcast schedules must be periodic [7], and the variance of spacing between consecutive instances of the same item must be reduced [8]. A number of push systems appeared, all nonadaptive. Here we focus on [6], which also produces periodic schedules (see [9]). It is based on the following arguments.

1) Schedules with minimum overall mean access time are produced when the intervals between successive instances of the same item are equal.
2) Under the assumption of equally spaced instances of the same items, the minimum overall mean access time occurs when the server broadcasts an item i with frequency being proportional to the factor $\sqrt{(d_i/l_i)((1+E(l_i))/(1-E(l_i)))}$, where d_i is the demand probability for item i, l_i is the item's length, and $E(l)$ is the probability that an item of length l is received with an unrecoverable error.

The algorithm operates as follows. Assuming that T is the current time, $R(i)$ is the time when item i was last broadcast, and the number of data items is M, the broadcast scheduler chooses to broadcast item i having the largest value of the cost function $G(i) = (T - R(i))^2(d_i/l_i)((1 + E(l_i))(1 - E(l_i)))$, $1 \leq i \leq M$. For items that have not been previously broadcast, $R(i)$ is set to -1. If the maximum value of $G(i)$ is shared by two or more items, the algorithm selects one of them arbitrarily. Upon broadcast of item i at time T, $R(i)$ is set to T and the algorithm proceeds to select the next item to broadcast. The above discussion assumes a single broadcast channel. However, an extension for operation in environments with more than one channel is also presented in [6].

The above algorithm is of $O(M)$ computational complexity. To reduce its complexity, a scheme known as bucketing divides the server's data items into queues (buckets), with each bucket containing items with close values of p_i/l_i. For more details, see [6].

A. The Motivation for Adaptive Push Systems

Existing push systems are useful only for static environments since no means for updating item probabilities is provided. Furthermore, pull and hybrid approaches have the disadvantages mentioned in Section I. Thus, it would be beneficial to reach a method that combines the advantages of the push and pull approaches—scalability and adaptivity, respectively. To this end, this paper enhances the nonadaptive method in [6] with a learning-automaton adaptation mechanism. To our knowledge, it is the first adaptive push-based system and provides both scalability and adaptivity. As will be seen, adaptivity comes at no expense over the computational complexity of [6] in the server side and the complexity of mobile clients' hardware. The only extra functionality demanded in the client side regards the transmission of a power-controlled feedback pulse.

III. The Adaptive Push System

A. Learning Automata

Learning automata [10] are mechanisms that can be applied to learn the characteristics of a system's environment. A learning automaton is an automaton that improves its performance by interacting with the random environment in which it operates. Its goal is to find among a set of M actions the optimal one, so that the average penalty received by the environment is minimized. This means that there exists a feedback mechanism that notifies the automaton about the environment's response to a specific action. The operation of a learning automaton constitutes a sequence of cycles that eventually lead to minimization of average penalty. The learning automaton uses a vector $p(n) = \{p_1(n), p_2(n), \ldots, p_M(n)\}$, which represents the probability distribution for choosing one of the actions $a_1, a_2, \ldots, a_M$ at cycle n. Obviously, $\sum_{i=1}^{M} p_i(n) = 1$.

The core of the operation of the learning automaton is the probability updating algorithm, also known as the reinforcement scheme, which uses the environmental response $\beta(n)$ triggered by the action a_i selected at cycle n to update the probability distribution vector p. After the updating is finished, the automaton selects the action to perform at cycle $n+1$, according to the updated probability distribution vector $p(n+1)$. A general reinforcement scheme has the form of (1)

$$\begin{aligned} p_i(n+1) =& p_i(n) - (1-\beta(n))g_i(p(n)) \\ &+ \beta(n)h_i(p(n)), \text{ if } a(n) \neq a_i \\ p_i(n+1) =& p_i(n) + (1-\beta(n))\sum_{j \neq i} g_j(p(n)) \\ &- \beta(n)\sum_{j \neq i} h_j(p(n)), \text{ if } a(n) = a_i. \end{aligned} \quad (1)$$

The functions g_i and h_i are associated with reward and penalty for action a_i, respectively, and $\beta(n)$ is a metric of the environmental response, normalized in [0,1]. The lower the value of $\beta(n)$, the more favorable the response. When $\beta(n)$ takes continuous values after normalization in the interval [0,1], the automaton is known as an S-model.

Learning automata have been applied to a number of problems, including the design of self-adaptive medium access control (MAC) protocols [11], queueing systems, image compression, pattern recognition, and telephone-traffic routing [10].

B. The Adaptive Scheduling Algorithm

The broadcast schedule construction of [6] assumes *a priori* knowledge of the client population demands for data items, which leads to inferior performance in cases of dynamic demands. However, the enhancement brought by the proposed adaptive broadcast scheduling algorithm enables broadcast schedule construction to adapt to changing demands of the client population, thus resulting in superior performance in these cases.

To this end, the proposed method suggests that the broadcast server uses a learning automaton whose probability distribution vector contains the server's estimate p_i of the actual demand d_i of the overall client population for each data item i. The parameter d_i is a metric that reflects the overall popularity of each item at the client population and actually stands for the probability that when an item request is made by the client population, this will be a request for item i. For example, assume a broadcast server having three items 1, 2, and 3 to broadcast and a client population making item requests. Requests will be

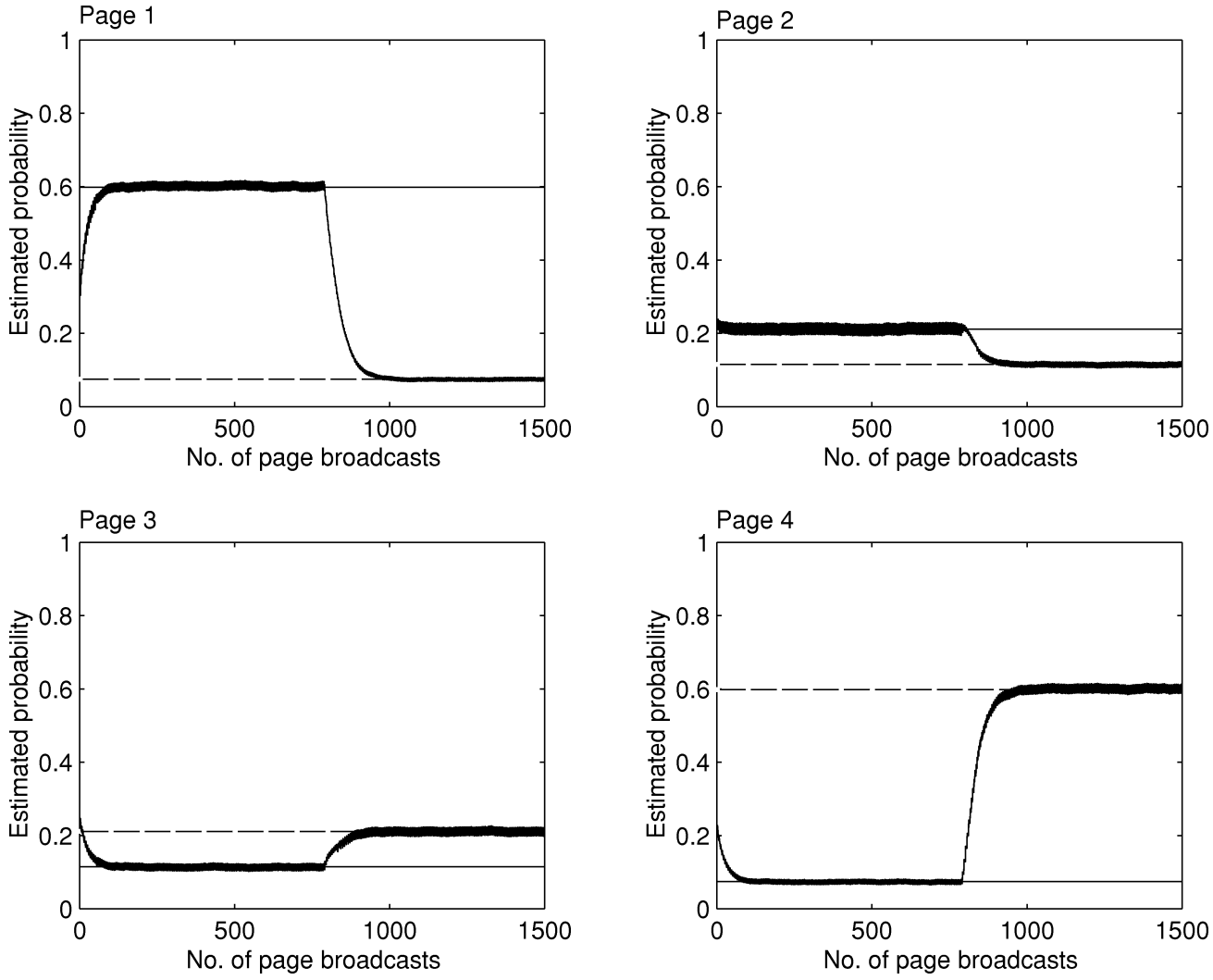

Fig. 1. Convergence of automaton estimation of the demand probabilities for pages one to four.

for one of items 1, 2, and 3. If the probabilities of a request's being a request for item 1, 2, or 3 are 60% 40%, and 0%, respectively, then $d_1 = 0.6$, $d_2 = 0.4$, and $d_3 = 0$. Obviously, the higher the d_i, the more requests will be made for item i and thus the more popular item i will be among the client population. Clearly, $\sum_{i=1}^{M} p_i = 1$ and $\sum_{i=1}^{M} d_i = 1$, where M is the number of items to be broadcast.

According to [6], the server chooses to broadcast page i having the largest value of the cost function G. Our approach extends this method: after broadcasting item i, the server waits for acknowledgment from all clients that were satisfied by this broadcast. A client that was waiting for item i acknowledges the reception via a short feedback pulse. The aggregate received signal strength at the server is obviously the environmental response for the broadcast of page i and will be used by the automaton to update the probability distribution vector p. It is noted that the adaptive broadcast scheduling algorithm takes into account transmission errors due to the use of $E(l)$ in function G.

However, the signal strength of each client's pulse at the server depends on its relative distance from the server and is of dynamic nature due to the mobility of the clients. If the path loss is a $1/x^n$-type loss with a typical $n = 4$, the feedback pulse of clients located close to the broadcast server will be orders of magnitude stronger than those of clients further away. To prevent the clients close to the server from dominating the voting, we introduce a power-control mechanism on the returning pulses. Thus, every page can be broadcast including information about the signal strength used for the page's transmission. Due to path loss, clients will receive the page and measure a lower signal strength at reception. Based on the information on the signal strength at which the page was originally transmitted S_0 and the signal strength S measured at the page reception (obviously $S \leq S_0$), clients will set the strength of their feedback pulse to S_0/S. Using this form of power control, the contribution of each client's feedback pulse at the server will be of the same order of magnitude, irrespective of the client–server relative position.

The probability distribution vector at the automaton determines the demand probability estimate of each information item. Using this scheme, the acknowledging clients' pulses add at the server and the automaton uses the strength of the received

pulse[2] to update the server's estimate of page probabilities. Thus, for the next broadcast, the server chooses which page to transmit by taking into account the updated values of the page probability estimates p_i. The way the automaton's estimation of page probabilities is updated through client feedback is described in Section III-C.

C. The Probability Updating Scheme

To provide adaptivity to the static method in [6], our approach suggests that the server use the probability updating scheme of an S-model linear reward minus inaction $(\mathrm{SL_{R-I}})$ learning automaton [10]. When an unfavorable response is received from the environment for the broadcasting of page i, the probability estimates of the pages do not change. Obviously, an unfavorable response results when the broadcast of a page does not satisfy any client. Following a favorable response, however, the probability estimate of page i is increased. The probability updating scheme of (2) is employed after the broadcast of page i (assuming it is the server's kth broadcast)

$$\begin{aligned} p_j(k+1) &= p_j(k) - L(1-\beta(k))(p_j(k)-a), \quad \forall\, j \neq i \\ p_i(k+1) &= p_i(k) + L(1-\beta(k)) \sum_{j \neq i} (p_j(k)-a). \end{aligned} \tag{2}$$

Equation (2) stems from (1) by setting $g_i(p(n)) = L(p_i(n)-a)$ and $h_i(p(n)) = 0$, where L is a parameter that governs the speed of the automaton convergence. The selection procedure for a value of L reflects the classic problem of speed versus accuracy. The lower the value of L, the more accurate the estimation made by the automaton, a fact however that comes at the expense of convergence speed. It holds that $L, a \in (0,1)$ and $p_i(k) \in (a,1), \forall\, i \in [1,\ldots,M]$, where M is the number of the server's pages. Should the probability estimate p_i of a page i become zero, then $G(i)$ would be very close to zero. However, the page, even if unpopular, still needs to be transmitted since some clients may request it. Moreover, the dynamic nature of client demands might make this page popular in the future. The role of parameter a is to prevent the probabilities of nonpopular pages from taking values in the neighborhood of zero in order to increase the adaptivity of the automaton. Upon reception of the sum of the acknowledging, power-controlled client pulses, this sum is normalized in the interval [0,1]. $\beta(k)$ represents the normalized environmental response after the server's kth broadcast. A value of $\beta(k)$ that equals one represents the case where no client acknowledgment is received. Thus, the lower the value of $\beta(k)$, the more clients were satisfied by the server's kth broadcast.

Using the reenforcement scheme of (2), the item probabilities estimated by the automaton converge near the actual demand probabilities for each page, making this approach attractive for dissemination applications with dynamic client demands. This convergence is shown in Fig. 1 for $L = 0.15$, $a = 10^{-4}$. In this figure, for each broadcast we plot the convergence of the page probability estimates toward the actual overall demand probabilities for pages one to four, in a simulation of a sample scenario comprising a database of four pages with lengths uniformly distributed in [1, ..., 10], 10 000 clients, and a wireless environment in which broadcast items are subject to errors at reception (the way that errors occur is described in Section IV). The overall client demands are initially unknown to the server. They are also of dynamic nature: At some time instant, the initial overall demand probability for each page (solid line) changes to a new one (dashed line). It is clearly seen that convergence of page probabilities estimated by the automaton to the overall client demand for these pages is achieved.

[2]The addition of the signal strengths is also mentioned in [12] as a way of distinguishing wireless transmission errors from packet collisions.

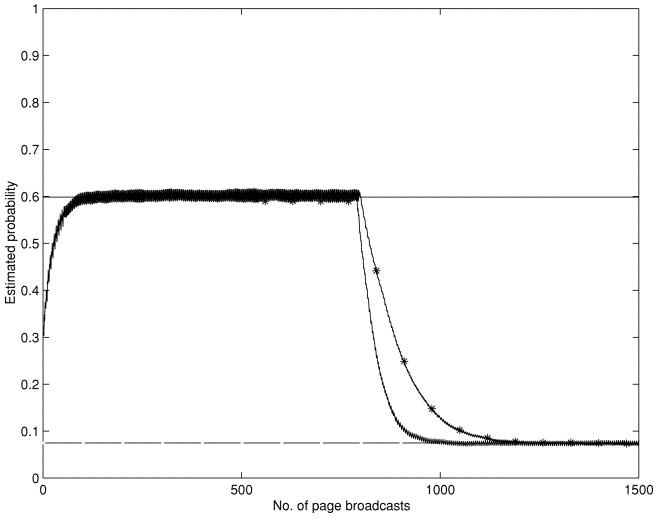

Fig. 2. Convergence of automaton estimation for page 1. The straight plot shows convergence when the number of clients between successive estimations is the same. The starred plot shows convergence in the case of the client population's being reduced by 50% after about 800 page transmissions.

To better understand the behavior of the reenforcement scheme, we detail the first probability update for the simulation of the above scenario. For the server's four pages $l_1 = 1$, $l_2 = 6$, $l_3 = 2$, $l_4 = 9$, $d_1 = 0.59$, $d_2 = 0.21$, $d_3 = 0.11$, $d_4 = 0.09$, and $p_1(1) = p_2(1) = p_3(1) = p_4(1) = 0.25$, where l_i is the length of page i, d_i is its actual overall probability of demand among the clients, and $p_i(k)$ is the server's estimate of the demand probability for page i to be used for the selection of the page for the kth broadcast. Furthermore, $R(i)$ denotes the time that item i was last broadcast. As stated in Section II, the values for $R(i)$ are initialized to -1. Since $G(i)$ is largest for $i = 1$, page 1 is broadcast and $R(1)$ is set to zero. The transmission of page 1 triggers a value of $\beta(1) = 0.55$. Using the reenforcement scheme of (2), the server updates its estimates of the pages' demand probabilities to $p_1(2) = 0.313$, $p_2(2) = 0.229$, $p_3(2) = 0.229$, and $p_4(2) = 0.229$, which will then be used for the selection of the page for the server's second broadcast.

The normalization procedure in the calculation of $\beta(k)$ implies that there must exist a mechanism that enables the server to possess an estimate of the number of clients under its coverage. This can be achieved by broadcasting a control packet, which forces every client in the cell to respond with a power-controlled feedback pulse. The server uses the aggregate incoming pulse E to estimate how many clients are within its coverage area. Then, upon reception of an aggregate feedback pulse of strength Q after the server's kth broadcast, $\beta(k)$ is calculated as Q/E.

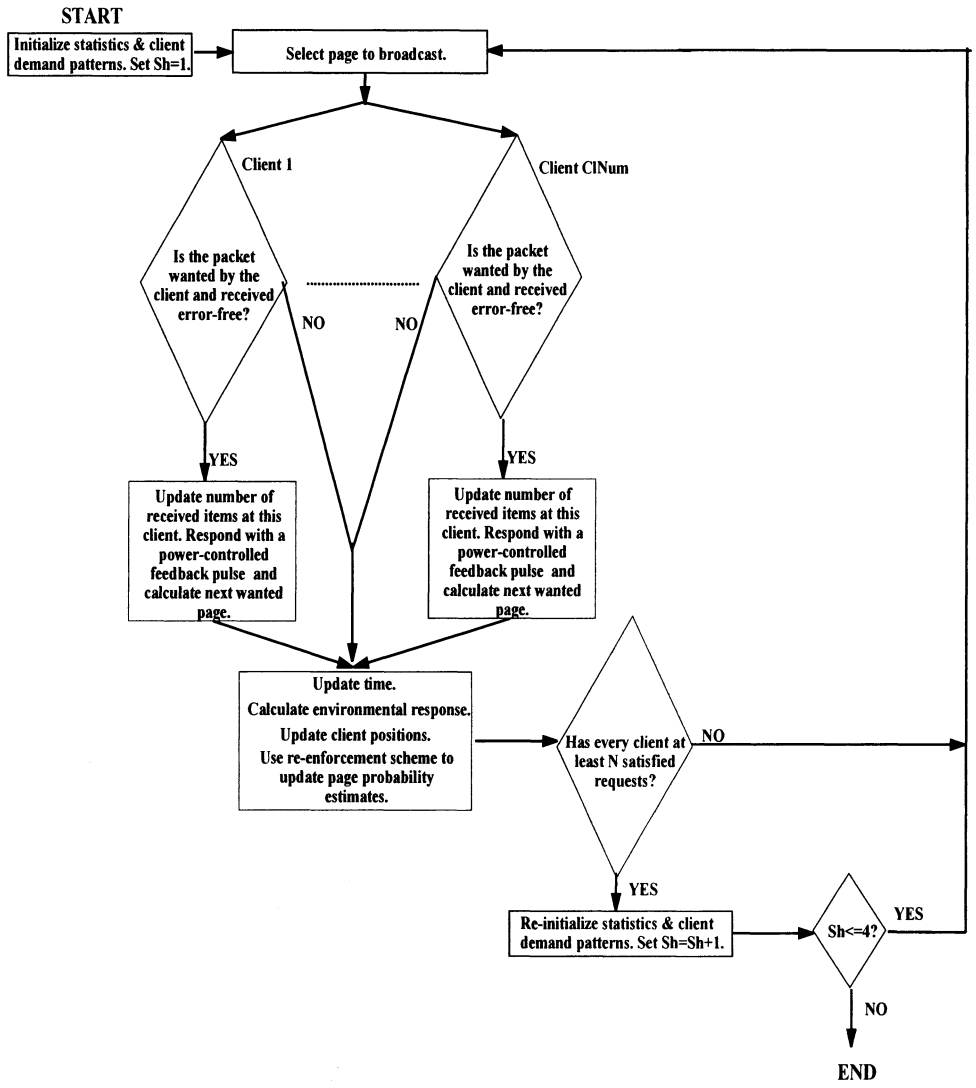

Fig. 3. Diagram of the simulation process.

The estimation process can occur at regular time intervals (e.g., after the broadcast of some hundreds of pages), with its overhead being the broadcast of a unit-length page. The simulation results in Section IV show that such an overhead is negligible due to the superior performance of the adaptive approach.

The estimation of clients at regular time intervals means that between two consecutive estimations, the broadcast server may for some time possess imprecise knowledge regarding the number of clients within the cell. Thus, we define that clients arriving at the cell are allowed to send feedback for received pages only after they have joined the process for the estimation of number of clients within the cell. This can be seen as a kind of registration aiming to maintain the precision of the environmental response in cases of an increasing number of clients between consecutive estimations. On the other hand, a decreasing number of clients between consecutive estimations leads to a lower value of $\beta(k)$ and thus $L\beta(k)$. This can be seen as a temporary reduction of the adaptation parameter L, leading to slower convergence of the automaton. To estimate the effect of this reduction on the protocol's performance, we plot in Fig. 2 the convergence of the page probability estimate toward the actual overall demand probabilities for page 1 in a simulation of the above-mentioned sample database scenario. In Fig. 2, two plots appear: The first (straight) shows convergence when the number of clients between successive estimations is the same. In the second plot (stars), just before the overall demand for page 1 lowers to about zero (dashed line), the client population is uniformly reduced by 50%. It can easily be seen that although at a slower rate, convergence to the overall demand probability of the remaining client population is still achieved.

The probability updating scheme of (2) is of $O(M)$ complexity. Therefore, the adaptive method maintains the $O(M)$ complexity of the static method in [6]. The bucketing scheme would not reduce complexity in the adaptive method, since due to the use of the probability updating scheme, complexity would remain $O(M)$.

IV. Performance Evaluation

Using simulation, we compared the proposed adaptive approach against the nonadaptive (static) approach of [6] in envi-

ronments characterized by unknown, dynamic client demands. In the static method, the server always broadcasts the entire range of its pages assuming equiprobable demand per page, since this is the best it can do without knowledge of the nature of overall client demands.

The reason for using the method of [6] for determining the efficiency of the adaptive method is twofold: a) determine its performance increase under dynamic client demands due to incorporation of adaptivity and b) compare against a pure push-based method and not against hybrid or pull methods, since we consider clients that do not have the ability of submitting requests to the server. This is not the case with pull and hybrid systems, which need a backchannel wide enough to submit data requests to the server.

A. Server Model

We consider a server that contains a database of M pages. The server is unaware of the client population demands, so initially every page has a probability estimate p_i of $1/M$. In the static method, the server always broadcasts pages assuming equiprobable demand, whereas in the adaptive method, page probabilities are updated according to the proposed scheme. Page lengths vary from $L_0 = 1$ to $L_1 = 10$, and two cases are considered [6]: a) random distribution, where page lengths are random integers uniformly distributed in $[L_0, \ldots, L_1]$, and b) increasing distribution, where the length l_i of a page i is $l_i = \text{round}(((L_1 - L_0)/(M-1))(i-1) + L_0)$, where round$(x)$ returns a rounded version of x.

B. Client Model

Clients are assumed to have no cache memory, as in [6]. Every client is initially set to access server pages in the interval [1, ..., Range], with Range $\leq M$. All pages outside this interval have a zero demand probability at the client. This page interval consists of an integral number of R regions with each region containing *RSize* pages. Pages inside the same region r have a same demand probability d_i of $d_i = c(1/r)^\theta$, where $c = 1/(\text{RSize}\sum_k (1/k)^\theta)$, $k \in [1, \ldots, R]$ and θ is a parameter named access skew coefficient. This is the Zipf distribution, used in other relevant papers as well [1], [6]. For small values of *RSize* together with large values of θ, the Zipf distribution produces increasingly skewed demand patterns and can thus model commonality in client demands.

To simulate some disagreement among the demands of different clients, we introduce the parameters *Dev* and *Noise*. For every client, a coin toss, weighted by *Dev*, is made. If the outcome of the toss states that the client is to deviate from the initial overall client demand, then a new demand pattern for this client is generated: with probability *Noise*, the demand probability of each page in the range of pages accessed by the client is swapped with that of another page selected uniformly from the interval $[1, \ldots, M]$.

C. The Simulation Environment

We performed our experiments with an event-driven simulator coded in C. The simulator models *ClNum* mobile clients, the broadcast server, and the server-client links as separate entities. Fig. 3 shows a diagram describing the simulation process. To simulate propagation loss, we assume that mobile clients are uniformly distributed inside a circular cell of radius R. Thus, the probability of a client's being at radius r is $2rdr/R^2$. We assume that the broadcast server's antenna is at the center of the circular cell and a path-loss model of $1/x^4$, where x is the distance between the client and the server's antenna. To simulate client mobility, we assume that clients remain in a position for intervals following an exponential distribution with mean D pages. After spending its time in a certain position, a client is placed in a new position at radius r with probability $2rdr/R^2$. Clients generate requests according to their demand patterns. The broadcast server schedules broadcasts according to the automaton's estimates of the page probabilities p_i and the cost function G, and updates its estimation of p_is after receiving the aggregate feedback pulse for the broadcast of an item. The pages broadcast are subject to reception errors, with unrecoverable errors per instance of a page occurring according to a Poisson process with rate λ, as in [6].[3] Thus, $E(l) = 1 - e^{-\lambda l}$ is the probability that a page of length l is received with an unrecoverable error. For the case of two broadcast channels, we assume that $E(l)$ is the same for both channels.

The simulation runs until at least N requests are satisfied at each client. Then, the demand per client changes, mimicking dynamic environments, and the simulation proceeds with the new demands active. This is repeated Sh times. The overhead due to the duration of the feedback pulse and the signal propagation delay is considered to be small compared to the page transmission time (parameter Ovh), as would happen in low-speed broadcasting applications confined in an area of several kilometers.

D. Simulation Results

The simulation results presented in this section were obtained with the following parameters values: $M = 500$, $ClNum = 10\,000$, $N = 2000$, $Sh = 4$, $Ovh = 10^{-3}$, $D = 100$, $Noise = 0.5$, $L = 0.15$, $\lambda = 0.1$, and $a = 10^{-4}$. Figs. 4–9 display simulation results for the case of a single broadcast channel. Fig. 10 displays results for the case of two broadcast channels. For page lengths following the random distribution, Figs. 4–6 display results for values of *Dev* being 0, 0.15, and 0.6, respectively. For page lengths that follow the increasing distribution, Figs. 7–9 do the same. In each of these figures, three pairs of plots appear, with each pair comparing the performance of the adaptive approach (dashed plots) to that of the nonadaptive approach (solid plots) for a different scenario. The three scenarios highlighted in the figures are N_1, with Range$= 100$, RSize$= 1$; N_2 with Range$= 100$, RSize$= 10$; and N_3 with Range$= 500$, RSize$= 1$.

Before discussing the performance characteristics of the adaptive scheme, we would like to explain those of the static scheme.

In the case of random length distribution (Figs. 4–6), we observe a decreasing delay of the static schemes for large values

[3]This assumption stems from the fact that although errors in wireless links are of bursty nature, in a typical broadcast schedule consecutive instances of the same item will be separated in time; thus errors in different instances of the same data item are independent. For more details, see [13].

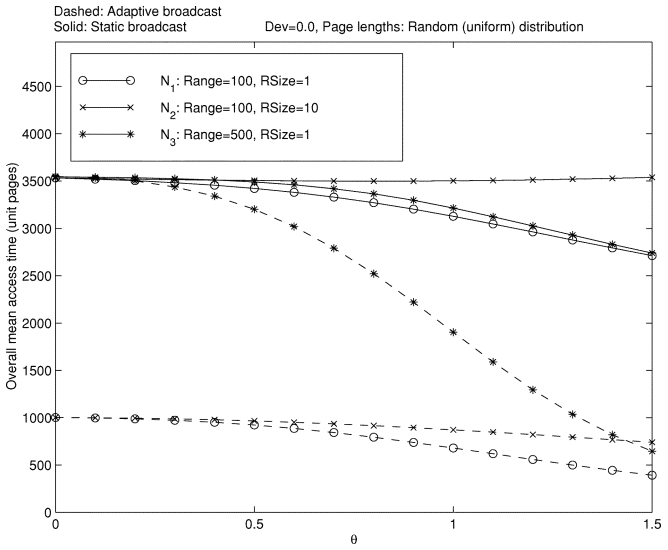

Fig. 4. Overall mean access time in unit pages versus access skew coefficient θ and using random length distribution (parameter *Dev* is set to 0.0).

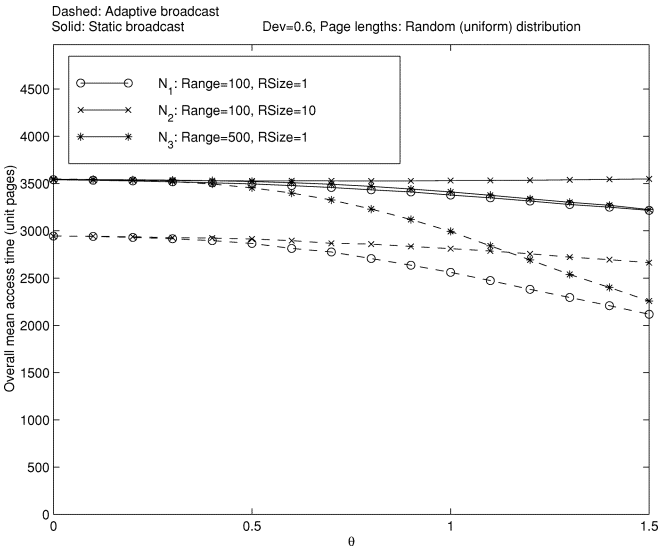

Fig. 6. Overall mean access time in unit pages versus access skew coefficient θ and using random length distribution (parameter *Dev* is set to 0.6).

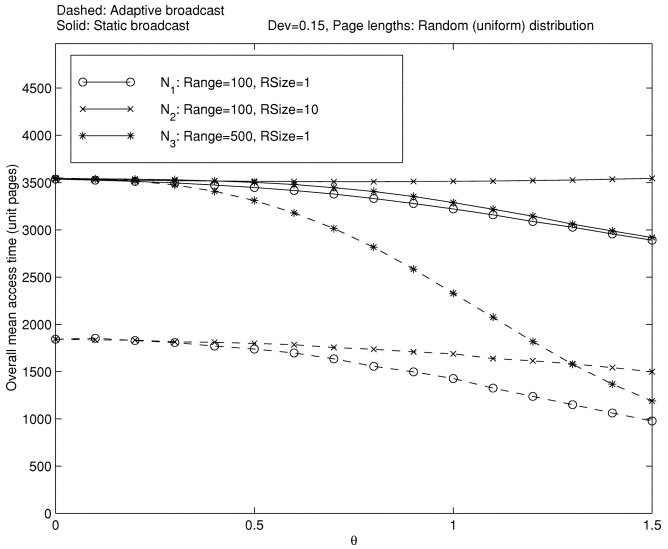

Fig. 5. Overall mean access time in unit pages versus access skew coefficient θ and using random length distribution (parameter *Dev* is set to 0.15).

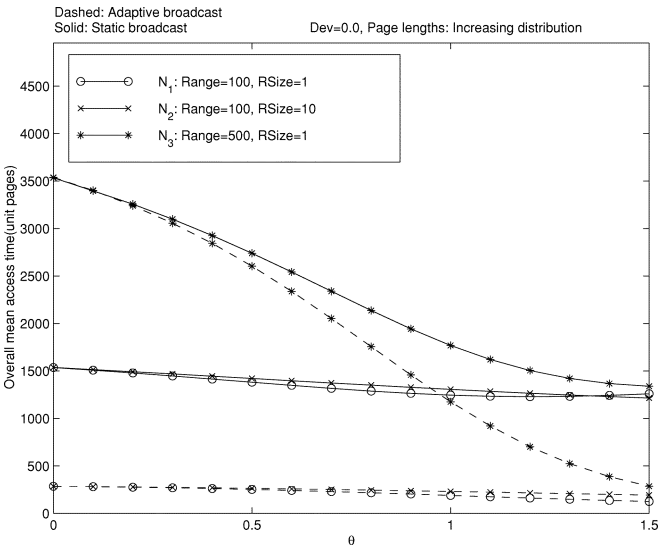

Fig. 7. Overall mean access time in unit pages versus access skew coefficient θ and using increasing length distribution (parameter *Dev* is set to 0.0).

of θ in N_1 and N_3. This is due to the fact that for large values of θ, the demand skew is so big that it makes the clients' demand probability for all pages practically zero, except for the demand for the first page. In our experiments with random page lengths, the length of the first page was one. In the case of N_1 and N_3 with Dev$=0.0$ and large θ, all clients select this page almost every time. Due to a) the small delay for page 1 transmission and b) the fact that pages appear in the broadcast with frequency inversely proportional to the square root of their length (see Section II), for large values of θ, clients in N_1 and N_3 generally have to wait less time to receive page 1, a fact that results in a decrease of the overall delay. For increasing values of *Dev*, this delay decrease for the static scheme is not so big, since for a percentage F of the client population $(F \leq Dev)$ the most popular page will be a larger one, due to the previously described swapping procedure used to introduce disagreement in client demands. The static scheme in N_2 is not subject to this performance decrease, since the use of RSize$=10$ produces a less skewed demand pattern. As a result, for large values of θ, clients in N_2 select among the first ten equiprobable pages and not only the first one. Since a random length distribution is used, some of these pages will be larger than the first one, yielding a nondecreasing delay.

In the case of increasing length distribution (Figs. 7–9), the decreasing delay of the static scheme for large values of θ in N_3 can be explained using the above reasoning. The same kind of reasoning can explain the difference in mean access time between N_3 and N_1, N_2 for small values of θ in the static case. Since we use an increasing length distribution, the use in N_1 and N_2 of Range$=100$ leads the clients to select among the first 100 pages of the server's database, which are of length of at most three. On the other hand, for small values of θ in N_3, all the pages in the server's database are almost equiprobably selected by the clients. Since, according to Section II, small-length

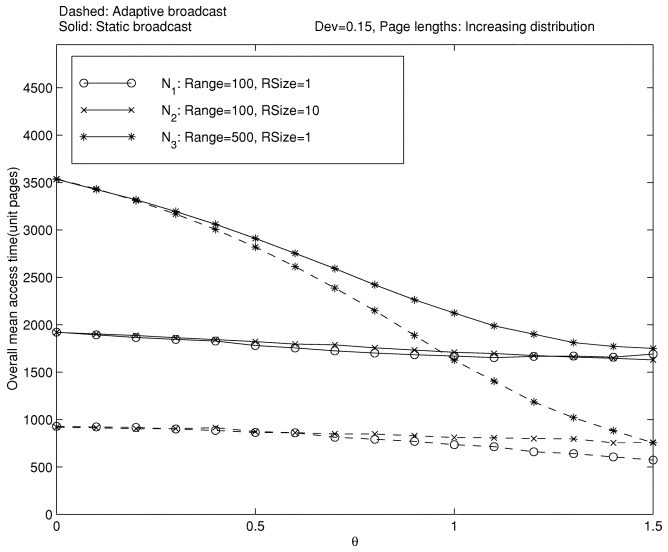

Fig. 8. Overall mean access time in unit pages versus access skew coefficient θ and using increasing length distribution (parameter *Dev* is set to 0.15).

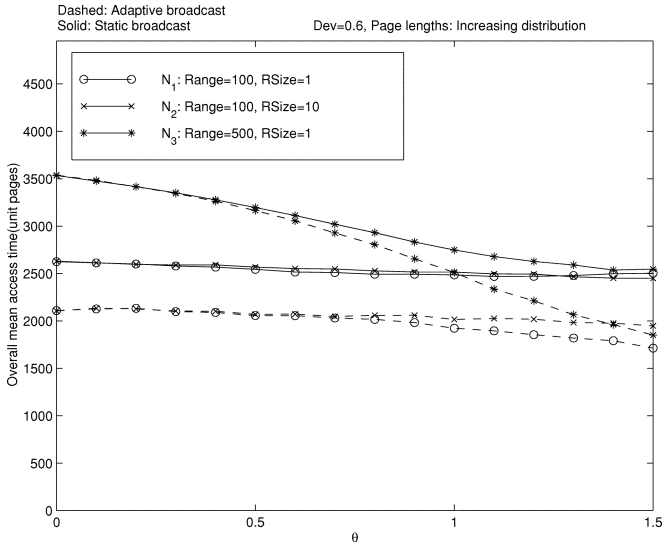

Fig. 9. Overall mean access time in unit pages versus access skew coefficient θ and using increasing length distribution (parameter *Dev* is set to 0.6).

pages are more frequently transmitted than larger ones, clients that select only among small pages will get more requests satisfied within a period of the static broadcast and consequently experience a lower access time. Furthermore, the delay for these pages' transmission is small. These arguments explain the low delay values of the static broadcast schemes in N_1 and N_2 for low values of θ. Such a behavior is not observed in the static schemes for random page-length distributions. This is because the first 100 pages are of lengths uniformly distributed in the interval [1, ..., 10] and not just of small lengths.

Regarding the adaptive scheme, we observe in Figs. 4–9 its superiority over the static one. The main conclusions that can be drawn from these results follow.

When all clients follow the same access pattern (Dev$= 0.0$; Figs. 4 and 7), the performance of N_1 and N_2 is at least 3.5 times better than that of the corresponding static scheme. This performance improvement decreases for increasing values of *Dev*. However, even in the case of Dev$= 0.6$ (Figs. 6 and 9), the adaptive schemes are in most cases at least 15% faster than the corresponding static schemes.

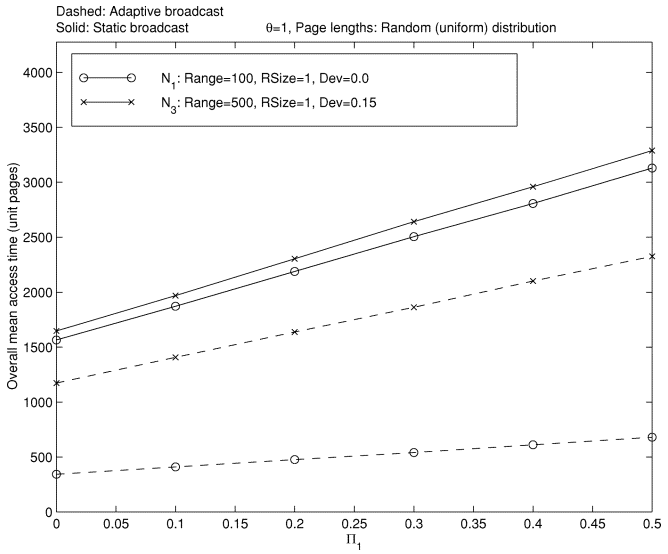

Fig. 10. Overall mean access time in unit pages versus the probability of listening to channel 1 Π_1 when using the random length distribution in the case of two available broadcast channels. $\Pi_1 = \Pi_2 = (1 - \Pi_{1,2})/2$.

Our scheme is useful even in cases of clients that access subsets of the server's database with demand patterns close to being uniform (small values of θ). This can be seen in Figs. 4–9 for N_1 and N_2, where Range$\leq M$. Even though this demand tends to be uniform for small θ, our scheme works best since it notifies the server to broadcast only the demanded items. For small values of *Dev*, the performance increase is bigger, as the swapping procedure will affect the demand of few clients. Thus, the increased commonality in client demands for pages in [1, ..., 100] increases performance. Clearly, the efficiency of the adaptive method for N_1 and N_2 exists for increasing values of θ as well.

For small values of θ, N_3's performance is close to that of the corresponding static scheme in Figs. 4–9. This is due to the fact that in N_3, $Range = M$ which means that clients pick up pages from the entire database of the server. Furthermore, this demand pattern for decreasing values of θ leads to an equi-probable probability distribution for the demand of the server's pages. Thus, for small θ, the adaptive broadcast schedule for N_3 tends to the static one produced for equi-probable pages. For increasing values of θ however, N_3's performance increases significantly.

We also simulated the static and adaptive scheme for N_1 and N_3 in the case of two available broadcast channels. For $\theta = 1$ we simulated N_1 and N_3 for *Dev* of 0.0 and 0.15, respectively. We assume that a client is capable of listening to channel 1, channel 2, or both of these channels with probabilities Π_1, Π_2, and $\Pi_{1,2}$, respectively. Obviously, $\Pi_1 + \Pi_2 + \Pi_{1,2} = 1$. We set $\Pi_1 = \Pi_2 = (1 - \Pi_{1,2})/2$ [6]. For various values of Π_1, the superiority of the adaptive scheme over the static schemes is clearly seen in Fig. 10. This superiority increases for increasing values of Π_1.

Based on the above discussions, the main observations that point out the significance of the proposed adaptive approach in terms of performance are the following.

1) In cases of applications that access the entire database of the server, the performance of the adaptive scheme is significantly improved over that of the static scheme, for increasing commonality in client demands (increasing θ and decreasing *Dev*).
2) The adaptive scheme is also superior to the static scheme, in cases of applications where clients access pages in subsets of the database, with demand patterns close to being uniform. Of course, this superiority also exists for all cases where clients access pages in database subsets with medium and highly skewed demand.
3) The performance efficiency of the adaptive scheme also holds when two broadcast channels are used.

V. Conclusion

This paper proposes an adaptive push system, which uses a learning automaton at the broadcast server. After an item's broadcast, each client waiting for this item acknowledges the reception via transmission of short, power-controlled feedback pulse. The acknowledging clients' pulses add at the server, and the automaton uses the strength of the received pulse to update the page probability estimates. These converge to the actual demand probability for each page, making this approach attractive for dissemination applications with dynamic client demands. The method is incorporated into a static push system [6]. Despite the added efficiency, it maintains the $O(M)$ complexity of [6]. Presented simulation results reveal efficient operation under dynamic client demands. In the future, we plan to investigate an extension of the adaptation method utilizing feedback pulses that arrive with different strengths at the receiver in order to support cases where client requests are subject to deadlines or applications with data items of different priorities.

References

[1] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-based data delivery using broadcast disks," *IEEE Pers. Commun.*, vol. 2, pp. 50–60, Dec. 1995.

[2] D. Aksou and M. Franlkin, "Scheduling for large-scale on-demand data broadcasting," in *Proc. IEEE INFOCOM*, 1998, pp. 651–659.

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

[4] J. Fernandez and K. Ramamritham, "Adaptive dissemination of data in time-critical asymmetric communication environments," in *Proc. 11th IEEE Euromicro Conf. Real-Time Systems*, 1999, pp. 195–203.

[5] T. Imielinski and S. Vishwanathan, "Adaptive wireless information systems," in *Proc. SIGDbS*, 1994.

[6] N. H. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments," in *ACM/Baltzer Wireless Networks*, 1999, vol. 5, pp. 171–182.

[7] M. H. Ammar and J. W. Wong, "On the optimality of cyclic transmission in teletext systems," *IEEE Trans. Commun.*, vol. COM-35, pp. 68–73, Jan. 1987.

[8] R. Jain and J. Werth, "Airdisks and airRAID: Modeling and scheduling periodic wireless data broadcast (extended abstract)," Rutgers—The State University, Piscataway, NJ, 1995.

[9] N. H. Vaidya and S. Hameed, "Data broadcast scheduling: On-line and off-line algorithms," Computer Science Department, Texas A&M Univ., 96-017 , 1996.

[10] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[11] G. I. Papadimitriou and A. S. Pomportsis, "Learning automata-based TDMA protocols for broadcast communication systems with bursty traffic," *IEEE Commun. Lett.*, vol. 4, pp. 107–109, Mar. 2000.

[12] P. Bhaqwat, P. Bhattacharya, A. Krishna, and S. Tripathi, "Enhancing throughput over wireless LAN's using channel state dependent packet scheduling," in *Proc. IEEE INFOCOM*, 1996, pp. 1133–1140.

[13] N. H. Vaidya and S. Hameed, "Improved algorithms for scheduling data broadcast," Computer Science Department, Texas A&M Univ., 96-029, 1996.

Petros Nicopolitidis received the B.S. and Ph.D. degrees in computer science from the Department of Informatics, Aristotle University of Thessaloniki, in 1998 and 2002, respectively.

His research interests are in the areas of wireless local area networks and mobile communications. He is coauthor of the book *Wireless Networks* (New York: Wiley, 2003).

Georgios I. Papadimitriou (M'89–SM'02) received the Diploma and Ph.D. degrees in computer engineering from the University of Patras, Greece, in 1989 and 1994, respectively.

From 1989 to 1994, he was a Teaching Assistant at the Department of Computer Engineering of the University of Patras and a Research Scientist at the Computer Technology Institute, Patras, Greece. From 1994 to 1996, he was a Postdoctorate Research Associate at the Computer Technology Institute. From 1997 to 2001, he was a Lecturer at the Department of Informatics, Aristotle University of Thessaloniki, Greece. Since 2001, he has been an Assistant Professor at the Department of Informatics, Aristotle University of Thessaloniki, Greece. His research interests include wireless networks, optical networks, and learning automata. He is coauthor of the books *Wireless Networks* (New York: Wiley, 2003) and *Multiwavelength Optical LANs* (New York: Wiley, 2003). He is the author of more than 70 refereed journal and conference papers.

Prof. Papadimitriou is a member of the Editorial Board of the *International Journal of Communication Systems*. He is also an Associate Editor of the journal *Simulation: Transactions of the Society for Modeling and Simulation International*.

Andreas S. Pomportsis received the B.S. degree in physics and the M.S. degree in electronics and communications from the University of Thessaloniki, Greece, and the Diploma degree in electrical engineering from the Technical University of Thessaloniki, Greece. In 1987, he received the Ph.D. degree in computer science from the University of Thessaloniki.

Currently, he is a Professor at the Department of Informatics, Aristotle University of Thessaloniki, Greece. He is coauthor of the books *Wireless Networks* (New York: Wiley, 2003) and *Multiwavelength Optical LANs* (New York: Wiley, 2003). His research interests include computer networks, learning automata, computer architecture, parallel and distributed computer systems, and multimedia systems.