Continuous Flow Wireless Data Broadcasting for High-Speed Environments

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

***Abstract*—With the increasing popularity of wireless networks and mobile computing, data broadcasting has emerged as an efficient way of delivering data to mobile clients having a high degree of commonality in their demand patterns. This paper proposes an adaptive wireless push system that operates efficiently in environments characterized by high broadcasting speeds and a-priori unknown client demands for data items. The proposed system adapts to the demand pattern of the client population in order to reflect the overall popularity of each data item. We propose a method for feedback collection by the server so that the client population can enjoy a performance increase in proportion to the broadcasting speed used by the server. Simulation results are presented which reveal satisfactory performance in environments with a-priori unknown client demands and under various high broadcasting speeds.**

***Index Terms*—Adaptive systems, data broadcasting, high-speed, learning automata.**

I. Introduction

DATA broadcasting [1]–[5] has emerged as an efficient means for the dissemination of information over asymmetric wireless networking environments [6]. Examples of data broadcasting are information retrieval applications, like traffic information systems, weather information and news distribution. In such applications, client needs for data items are usually overlapping. As a result, broadcasting stands to be an efficient solution, since the broadcast of a single information item is likely to satisfy a (possibly large) number of client requests.

Communications asymmetry is due to a number of facts:

- Equipment asymmetry: A server usually has powerful transmitters that are not subject to power limitations, whereas client transmitters are usually hindered due to finite battery life. Moreover, it is desirable to keep the mobile clients' cost low, a fact that sometimes results to lack of client transmission capability.
- Network asymmetry: In many cases, the available bandwidth for downlink transmission is much more than that in the opposite direction. Furthermore, there exist extreme network asymmetry cases, where the clients have no available backchannel. Even in the case of backchannel existence, the latter is subject to becoming a bottleneck in the presence of a very large client population.
- Application asymmetry: This is due to the pattern of information flow. Information retrieval applications are of client-server nature. This means that the flow of traffic from the server to the clients (downlink traffic) is usually much higher than that in the opposite direction (uplink traffic).

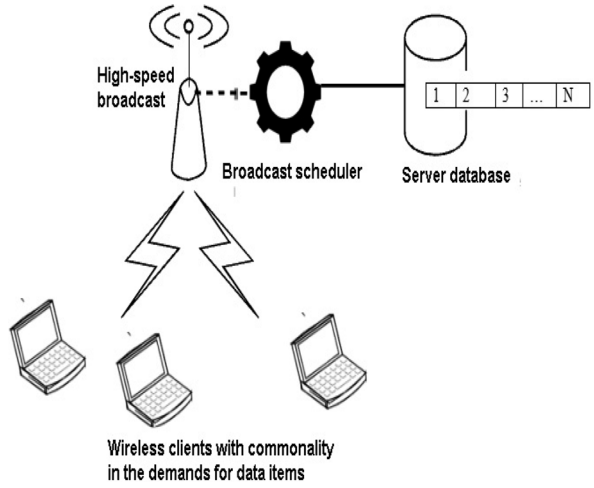

Fig. 1. System environment.

Until now, research on push-based systems assumed a-priori and static knowledge of the demand pattern. However today's information retrieval applications are characterized by demand patterns that are likely to be unknown and change with time. Consider a hypothetical scenario in an airport. 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 client demand. For a specific flight, the demand is likely to be in its peak a couple of hours before the flight 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 5 or 6 hours before their flight. At this time, the server should increase the frequency of data items concerning flights leaving in the near future. As the time for the departure approaches, the demand for information regarding our flight will grow due to the increasing number of waiting passengers, and eventually, after a few minutes of the departure of the flight, it will drop again. It can be easily seen that in such an environment the server needs to broadcast information according to the state of the client demand, which is neither a-priori known, nor is it static.

The environment in which the proposed system in this paper operates is shown schematically in Fig. 1. It can be seen that such an environment comprises a number of wireless clients

Manuscript received November 19, 2008; revised January 22, 2009. First published April 21, 2009; current version published May 22, 2009.

The authors are with the Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece. (e-mail: petros@csd.auth.gr).

Digital Object Identifier 10.1109/TBC.2009.2015982

with high commonality in their demand patterns. The clients are also capable of limited uplink communication. It also comprises a database server whose data items, after proper scheduling by the scheduler module, are delivered to the wireless clients via high-speed wireless data broadcasting. To provide efficient operation for an adaptive wireless push system in environments with high-speed broadcasting servers, the proposed approach enhances the adaptive push-based system of [7]. It uses a Learning Automaton at the server, which continuously adapts to the demand pattern of the client population in order to reflect the overall popularity of each data item. The adaptation is accomplished using a feedback from the clients. Using this approach, information items are transmitted according to client demands, which can be initially unknown to the server and time varying. The feedback is collected by the server that uses it to update the item probability estimation vector via the Learning Automaton. The item probabilities estimated by the Automaton converge near the actual overall item demand probabilities of the client population, making this approach attractive for dissemination applications with a-priori unknown client demands for data items.

The method to convey the client feedback to the server uses orthogonality in the code domain which, as explained later on, imposes a constraint on the minimum time needed to collect client feedbacks. The latter fact imposes performance limits when the adaptive push system that has been proposed in [7] operates in environments characterized by high server broadcasting speeds, as in such cases, the time period for feedback collection for each item that was broadcast is a performance bottleneck to the system. Thus, contrary to [7], we propose an in-band channel realized over the transmitted data items, over which the server probes the data items in order to estimate their probability of being demanded by the client population. The frequency of probing is selected in such a way so that a continuous flow of broadcast items is ensured and the performance of the system can scale according to the broadcasting speed of the server.

The remainder of this paper is organized as follows: Section II discusses related work in the area of push systems. Section III presents the proposed system and discuses how probability updating is performed via client feedback. Section IV presents simulation results, which reveal satisfactory performance in environments with a-priori unknown client demand patterns and under various high broadcasting speeds. Finally, Section V summarizes and concludes the paper.

II. Related Work

A. Push-Based Data Broadcasting

There are many papers addressing the problem of efficient data broadcasting. In what follows, we outline some of them. A popular approach in the area of push systems was the Broadcast Disks model [8]. It proposes a way of superposition of multiple disks spinning at different frequencies on a single broadcast channel. The most popular data are placed on the faster disks and as a result, periodic schedules were produced with the most popular data being broadcast more frequently. This work also proposes some cache management techniques aiming to reduce performance degradation of those clients with demands deviating from the server's schedule. This work was augmented later by dealing with issues such as efficient cache management based on prefetching [9], impact of changes at the values of the data items at the server between successive broadcasts [10] and addition of a backchannel to allow users to send explicit requests to the server [11]. The latter approach can be considered as a hybrid system. However, the Broadcast Disks approach is not adaptive, since it is based on the server's knowledge of a static client demand pattern resulting in pre-determined broadcast schedules. Furthermore, it is constrained to fixed sized data items and does not present a way of determining neither the optimal number of disks to use nor their respective frequencies. Those numbers are selected empirically and as a result, the server may not broadcast data items optimally, even in cases of static demand patterns.

Push-based systems are also proposed in [12]. This work proposes broadcast schedules based on the so-called square-root rule. Assuming that the instances of each item are equally spaced in the broadcast, it shows that the access time is minimized when the server broadcasts an item i with frequency proportional to the square root of the factor $p_{\mathrm{i}}/l_{\mathrm{i}}$, where p_{i} is the overall client demand probability for item i and l_{i} is this item's length. In [13] the same result is reached for equal l_{i} 's through numerical experiments and also examines the outcoming algorithms performance on a pull system. This work is carried to a further extent in [14], where it is argued that in addition to the mean access time, the variance of the response time should also been taken into account by the broadcast scheduler. The proposed algorithm provides a balance between minimization of the mean access time and the variance. In [15], the same problem is reviewed by taking into account deadlines for client requests.

B. The Adaptive Data Broadcasting System

The above described push-based approaches are practical only for fairly stable environments since no mean of updating the probabilities of data items is proposed. The Broadcast Disks approach is based on the server's knowledge of a static demand pattern resulting in pre-determined, non-adaptive, broadcast schedules. [12] and [13] assume that the access probabilities of data items are available and are thus limited in terms of robustness to changing demand. The method proposed in [7] proposes a push-based system that is adaptive to dynamic client demands via use of a Learning Automaton at the broadcast server. The Learning Automaton updates a probability distribution vector that contains the server's estimate of the actual demand of the overall client population for each data item. The demand for each item is a metric that reflects the overall popularity of the 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 that item.

The reason behind the use of a Learning Automaton at the broadcast server is the fact that it can provide a) a means of

learning the demand probability of each data item among the client population and b) does not increase the computational complexity of the non-adaptive system of [12] due to the fact that both the learning algorithm and the system of [12] are of linear computational complexity in regard to the number of items in the server's database [7]. Thus, the use of a Learning Automaton is an efficient means to provide both adaptivity to the broadcasting system as well as maintain the computational complexity of the broadcast scheduler.

In the system proposed in [7], via using 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. For every item broadcast, the server selects to broadcast the item having the largest value of the cost function proposed in [12]. Extending this, in the system proposed in [7], after broadcasting an item, the server waits for acknowledgment from all clients that were satisfied by this broadcast. All clients that were waiting for the broadcast item, acknowledge the reception via a feedback pulse and the aggregate feedback is used at the server by the automaton to update the probability distribution vector that reflects the overall popularity of each data item. This is done via a learning algorithm, of linear complexity in regard to the number of items in the server's database, an algorithm that is also used in the system proposed in this paper.

It is shown via simulation in [7] that, contrary to the method of [12], the adaptive system provides superior performance in an environment with a-priori unknown client demands, which can also change over time with the nature of these changes being unknown to the broadcast server. Further extensions of [7] address performance improvement [16], [17] and fairness [18] in environments with locality of client demands.

III. The Proposed Wireless Push System

A. Learning Automata

Learning Automata [19] 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 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 about the environment's response to a specific action. The operation of a Learning Automaton constitutes a sequence of time cycles that eventually lead to minimization of average penalty. The Learning Automaton uses a vector $p(n) = \{p_1(n), p_2(n), \ldots, p_A(n)\}$, which represents the probability distribution for choosing one of the actions $a_1, a_2, \ldots, a_A$ at time cycle n. Obviously $\sum_{i=1}^{A} p_i(n) = 1$. 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 α_i selected at cycle n to update the probability distribution vector p. After the update is finished ,the automaton selects the action to perform at time cycle $n+1$, according to the updated probability distribution vector $p(n+1)$. A general reinforcement scheme has the form of the following formula:

$$\begin{aligned} p_i(n+1) &= p_i(n) - (1-\beta(n))\, g_i\,(p(n)) \\ &\quad + \beta(n) h_i\,(p(n))\,, \qquad \text{if } \alpha(\mathrm{n}) \neq \alpha_{\mathrm{i}} \\ p_i(n+1) &= p_i(n) + (1-\beta(n)) \sum_{j \neq i} g_j\,(p(n)) \\ &\quad - \beta(n) \sum_{j \neq i} h_j\,(p(n))\,, \quad \text{if } \alpha(\mathrm{n}) = \alpha_{\mathrm{i}} \end{aligned} \tag{1}$$

The cycle n is defined as the time period in which the Learning Automaton chooses one of the actions $a_1, a_2, \ldots, a_A$, executes it, receives the environmental response $\beta(n)$ triggered by the action α_i and updates the probability distribution vector p. The functions g_i and h_i are associated with reward and penalty for action α_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. In the area of data networking Learning Automata have been applied to several research issues, including the design of self-adaptive MAC protocols for wired and wireless platforms which efficiently operate in dynamic workloads (e.g. [20]–[23]).

B. The Broadcasting Algorithm

To optimize performance, broadcast schedules must be periodic [24], and the variance of spacing between consecutive instances of the same item must be reduced [25]. Based on the above, the broadcast scheduling of many push systems [12e.g.] are based on the following arguments:

- Schedules with minimum overall mean access time are produced when the intervals between successive instances of the same item are equal.
- 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 $\sqrt{(p_i/l_i)((1+E(l_i))/(1-E(l_i)))}$, where p_i is the demand probability for item i, l_i is the item's length, and $E(l_i)$ is the probability than an item of length l_i is received with an unrecoverable error.

[12] reveals that a broadcast algorithm based on the above arguments minimize the mean response time of the system (optimize performance). Thus, the broadcasting algorithm used in this paper is also based on the above arguments and operates as follows: The server is equipped with an S-model Learning Automaton that contains the server's estimate p_{i} of the demand probability d_{i} for each data item i among the set of the items the server broadcasts. The estimation probability vector p stores the server's estimation of the actual demand probability vector d that contains the actual choice probabilities of the various information items averaged over the entire client population. Clearly $\sum_{i=1}^{N} p_i = \sum_{i=1}^{N} d_i$, where N is the number of items in the server's database. At each cycle, the server selects to transmit the item i that maximizes the cost function $G(i) = (T - R(i))^2 (p_i/l_i)((1+E(l_i)/1-E(l_i)))$, where T is the current time, $R(i)$ the time when item i was last broadcast, and as previously mentioned, l_{i} is the length of item i and $E(l_{\mathrm{i}})$ is the probability that an item of length l_{i} is erroneously received. For

items that haven't been previously broadcast, R is initialized to -1. If the maximum value of $G(i)$ is shared by more than one item, the algorithm selects one of them arbitrarily. Upon the broadcast of item i at time T, $R(i)$ is changed so that $R(i) = T$.

As mentioned in the introductory section, the server will probe the data items in order to receive feedback from them so as to estimate the demand probability for each data item. For the feedback transmission, CDMA is chosen [18]. Each client that was waiting for the data item being probed sends its feedback (i.e. one bit) to the base station using a user-specific high-speed code (Long code). At the receiver end (base station), signals are separated by using a correlator, which only accepts signals energy from the specific client's long code and despreads its spectrum. Other co-user signals remain spread because their spreading algorithm is uncorrelated with the desired signal's algorithm and they appear as noise. This method for feedback has the advantage of the same overhead for the server regardless the number of clients that are sending feedback for a specific item. This is due to the fact that the feedback by each client does not carry any information and the server is merely interested in its presence or absence. Thus, what the server always does is to just count the number of incoming feedbacks by always checking for feedback presence among CDMA feedback channels whose total number is constant for certain system parameters, as described in the next paragraph.

The CDMA technique has been preferred because CDMA has advantages in several aspects compared to other techniques [26], [27]:

- High capacity. The primary purpose of using CDMA is to achieve high capacity, which in our case translates to a larger number of supported mobile clients. CDMA operates on a common frequency carrier which can result to a frequency reuse factor of one. On the other hand, CDMA can tolerate noise to a great extend due to its wide bandwidth. All these factors contribute to the high capacity of CDMA system [26] which is given by the equation below:

$$P = \frac{W/U}{E_b/N_o} + 1 \tag{2}$$

  where P is the number of mobile clients supported, W represents the transmission bandwidth for the feedback, U represents the user transmission bit rate and E_b/N_o is the bit energy-to-noise power spectral density. Suggestively, for $W = 5\,\mathrm{MHz}$, $U = 1024\,\mathrm{bps}$ and $E_b/N_o = 7\,\mathrm{dB} = 5$, a population of $P = 977$ clients is calculated. In case there is need for more clients, the implementation of CDMA in more than one frequency channels can increase the required capacity.
- Noise suppression. The CDMA receiver can operate in very low SNR environment, due to its high processing gain.
- Low RF transmit power. The CDMA transmitters always transmit at the lowest possible power level. This helps save mobile battery life.

It has to be stressed that since data need to be transmitted at very low bit rates, due to the small amount of information that each client needs to send as a feedback (practically one bit coded with its Long Code), a simple implementation of a CDMA coding is considered more than enough to guarantee correct reception of what is sent by the clients. Moreover, although CDMA has the potential to deal with fading due to multipath effects it is expected that such effects will have minimal (if any) effects on the signal, due to the choice of a sufficiently low bit rate. The multipath effect limits the maximum bit rate, since multiple signals that are diffracted by various obstacles add up with different propagation delays. The maximum allowed bit rate is given by $U_{max} = 1/(2DS)$, where DS is the delay spread that characterizes multipath in a specific propagation environment. Typical values of the delay spread are 0.2 μsec for propagation in irregular terrain, 0.5 μsec for a suburban environment and 3 μsec for urban environment. With a worst value of 5 μsec, the maximum bit rate is 100 Kbps, hence a choice of a few Kbps is enough to guarantee full suppression of multipath effects.

Using (2), one can obtain the number of mobile clients that can be supported by a broadcast server. By keeping in mind that in data broadcasting systems we need to provide support for as many mobile clients as possible, it becomes obvious by (1) that the user transmission rate U must be kept at the lowest possible number, with values around 1024 bps indicated by various commercial CDMA standards (e.g. cdmaOne).

The above mentioned requirement will in turn translate to a need for the highest possible duration for the feedback transmission by the clients. However, although it does not cause problems in low-speed broadcasting environments, this requirement will pose a performance bottleneck for the method of [7] when the server broadcasts at high bit rates. This is due to the fact that in such high-speed environments, the ratio of packet transmission time to the feedback transmission time becomes very low, thus the server underutilizes the downlink broadcast channel since it is required to wait for the reception of feedback after each item broadcast. Thus, we propose, that contrary to the method of [7], which a) waits to collect feedback after the transmission of each item by the server and b) states that the collected feedback concerns the data item that was just transmitted, the feedback collection method of the proposed approach is differentiated. Specifically, in the proposed method the feedback a) is not transmitted by the clients after each broadcast item but rather upon request by the server and b) when requested by the server, the feedback that is received does not necessarily relate to the most recent data item transmission. In the following of this section we described the details of the proposed approach, which is also depicted via a flowchart in Fig. 2.

To achieve the requirements in the previous paragraph, we use a space of $1 + \log_2 N$ number of bits on the header of each broadcast item i, $1 \leq i \leq N$. By piggybacking on this space a value representing decimal j, $1 \leq j \leq N$, the server will request feedback by the clients that are currently waiting to receive item j. When the server piggybacks on the above mentioned space a value of zero, it indicates that it does not request feedback (thus it does not probe for any item j) after the broadcast of item i.

As far as the details of the probing mechanism are concerned, after the transmission of a probing request, the server will continuously (e.g., without probing requests) broadcast items for an aggregate time duration equal to at least the duration of the feedback transmission plus the maximum propagation delay from the server to the clients. Both these parameters are known a-priori since the feedback uses one user bit and the user transmission speed U is known. Moreover, the maximum

propagation delay is tied to the server transmission range and is also known. Thus, when the time duration since the last probing request exceeds the sum of the feedback transmission time and the maximum propagation delay from the server to the clients, the server will transmit the next item i in the schedule with a piggybacked probe for another item j. In order to select the item to probe, j, we separately apply the cost function $G(j)$ to produce the sequence of probed items by assuming that time elapses only when transmission of items with probing requests are made. This of course entails that the server needs to store a separate vector R' that stores the time when each item was last probed by the server.

Using the procedure described above, the server will at regular time intervals demand feedback from each client that was waiting for reception of the probed data item. The aggregate feedback is then used at the server to update the Learning Automaton. The probability distribution vector p maintained by the Automaton estimates the demand probability d_{i} (and thus the popularity) of each information item i. Until the next probability update takes place (which is obviously triggered by the next request for feedback) the server will use the updated vector p to calculate the cost G of each item in the process of choosing which one to broadcast.

When the probing of an item j does not yield client feedback due to the fact that no client was waiting for j, the probabilities of the items do not change. However, following a probe for j that yields client feedback the probability of j is increased. The following Liner Reward-Inaction ($\mathrm{L_{R-I}}$) probability updating scheme [19] is employed after the probing of item j (assuming it is the server's kth probing request):

$$\begin{aligned} p_m(k+1) &= p_m(k) - L\,(1-b(k))\,(p_m(k)-a), \quad \forall\, m \neq j \\ p_j(k+1) &= p_j(k) + L\,(1-b(k)) \sum_{m \neq j} (p_m(k)-a) \end{aligned} \tag{3}$$

where $p_j(k)$ takes values in $(\alpha, 1)$ and L, α take values in (0,1). The role of parameter α is to prevent the probabilities of non-popular items from taking values very close to zero in order to increase the adaptivity of the Automaton. This is because if the probability estimate p_{j} of an item j approaches zero, then $G(j)$ will take a value very close to zero. However, item j, even if unpopular, still needs to be transmitted since some clients may request it. Furthermore, the dynamic nature of client demands might make this item popular in the future. $b(k)$ represents the environmental response after the server's kth probing request. Upon reception of the client feedbacks, this sum is normalized in the interval [0, 1]. A value of $b(k)$ that equals 1 represents the case where no client feedback is received. Thus, the lower the value of $b(k)$, the more clients were satisfied by the server's kth probing request.

Using the probability updating scheme of (3), the item probabilities estimated by the Automaton converge near the actual demand probabilities for each item. This makes this approach attractive for dissemination applications with dynamic client demands. This convergence is schematically shown in Fig. 3, which plots the page probability estimates versus the overall actual demand probabilities, in a simulation of broadcasting a sample database comprising four pages and 2000 clients. The client demand pattern is unknown to the server, so initially, the four pages have an equal demand probability estimate It is clearly seen, that convergence of the page probabilities estimated by the Automaton to the actual overall demand pattern of the client population is achieved.

The probability updating scheme of (3) is of O(N) complexity. Therefore the adaptive push method does not increase the computational complexity of the non-adaptive one. The bucketing described in [12] would not further reduce computational complexity in the adaptive method, as complexity would remain of O(N) due to the probability updating scheme.

The normalization procedure in the calculation of $b(k)$ suggests the existence of a procedure to enable the server to possess an estimate of the number of clients under its coverage. This is made possible by broadcasting a control packet that forces every client in the cell to respond with a feedback. The broadcast server will use the total received feedback S to estimate the number of clients under its coverage. Then, in subsequent probing requests, upon reception of feedback from Z clients after the server's kth probing request, $b(k)$ is calculated as $1 - Z/S$. This estimation process will take place at regular time intervals with the negligible overhead of broadcasting a unit-length control item.

IV. Performance Evaluation

Using simulation, we compared the proposed system against the one of [7] and the non-adaptive broadcast scheme of [12], which schedules the items equiprobably. We consider $ClNum$ mobile clients and a broadcast server having a database of Dbs items. The server is initially unaware of the demand for each item, so initially every item has a probability estimate of $1/Dbs$. Page lengths vary from $L0 = 1$ to $L1 = 5$ and are set according to a uniform random distribution, where page lengths are random integers uniformly distributed in $[L0 \ldots L1]$. Every client accesses pages in the interval $[[1, Range]]$. This page range consists of R regions of size $RSize$ each, that contain pages with same demand probabilities. The demand probability $d(i)$ of a page in region i, is $d(i) = c(1/i)^{\theta}$ where $c = 1/Rsize \sum_k (1/k)^{\theta}$, $k \in [1 \ldots R]$ and θ is a parameter named access skew coefficient. This is the Zipf probability distribution used in other papers as well. θ is a parameter named access skew coefficient. For $\theta = 0$, the Zipf distribution reduces to the uniform distribution. As the value of θ increases, the Zipf distribution produces increasingly skewed demand patterns. The Zipf distribution can thus efficiently model applications that are characterized by a certain amount of commonality in client demand patterns.

To simulate some disagreement in the client demand patterns, we introduce the parameters Dev and $Noise$. These parameters determine the percentage of clients that deviate from the initial demand pattern described above and the degree of that deviation. 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 demand pattern, then a new demand pattern for this client is generated. This pattern is produced in the following way: with probability $Noise$ the demand probability of each page in the client's demand pattern database is swapped with another page that is selected in a uniform manner from the interval $[1 \ldots DbS]$.

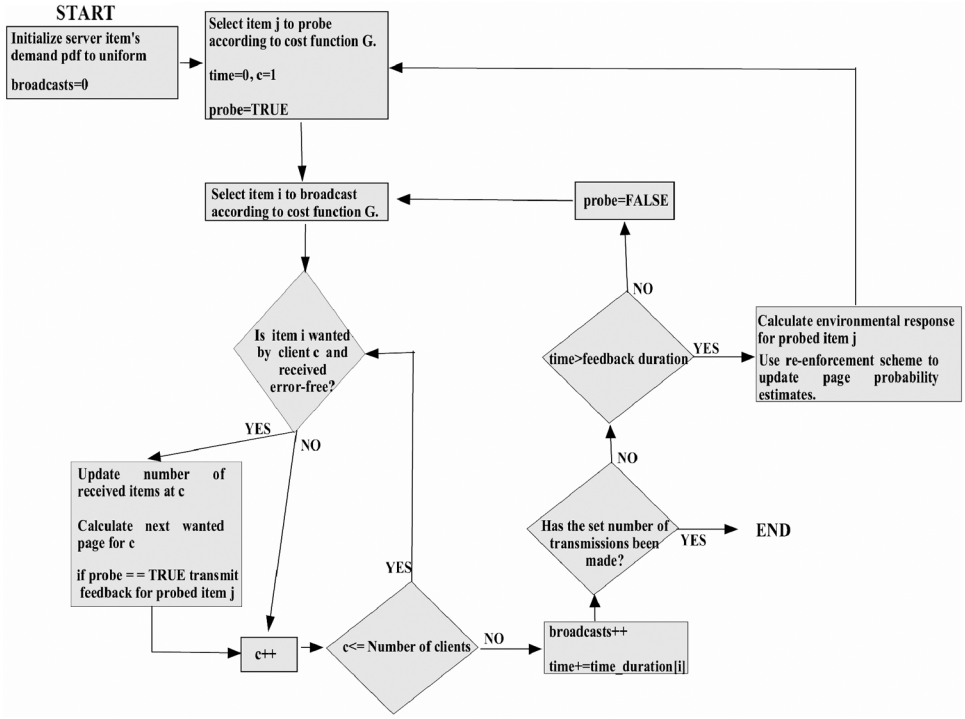

Fig. 2. Flowchart of the proposed approach.

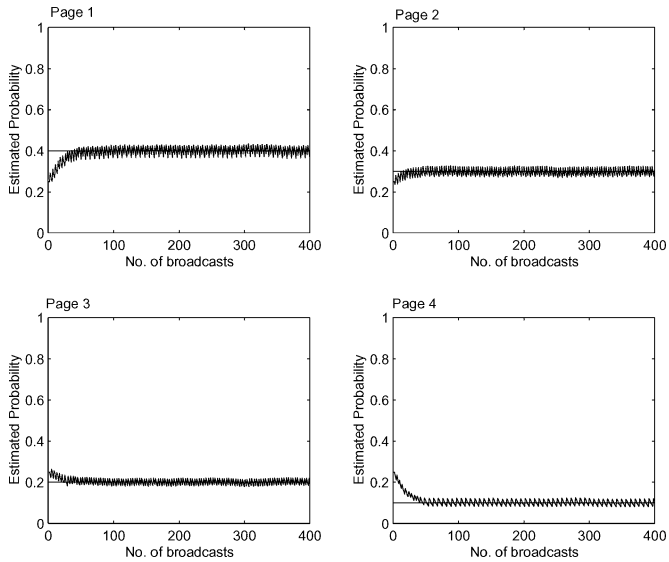

Fig. 3. Convergence of automaton estimation of the demand probabilities.

Each client generates requests according to its demand pattern. Upon reception of the requested page from the broadcast, the client generates another request and waits for its fulfillment. The simulation is carried out until at least $NumItems$ requests are satisfied at each client, meaning that overall, at least $ClNum * NumItems$ requests have been served.

The results were obtained with the following parameter values being constant:

- $DbS = 200$.
- $ClNum = 2000$.
- $NumItems = 2000$.
- $Noise = 0.5$
- $L = 0.15$.
- $\alpha = 10^{-3}$.
- Unit item size $= 10^3$ bits.
- $U = 1$ Kbps.
- $W = 15$ MHz.
- $Eb/No = 5$.

The propagation delay is considered negligible compared to the duration of the feedback transmission, as would be the case in situations with system deployment in cells of a few Km radius.

Figs. 4–11 contain simulation results for the following two network environments:

- Network N1: $Range = 200$, $RSize = 1$.
- Network N2: $Range = 50$, $RSize = 2$.

In the first environment, Network N1, the clients access the entire range of the data items $(Range = Dbs)$ in the server's database, while in the second one, Network N2, the clients access only the first quarter $(Range = Dbs/4)$ of the server's data items. The above difference results to a more skewed overall demand for Network N2, even in cases of a zero value of the access skew coefficient θ, which will only translate to a uniform demand for the first quarter of the server's data items that are actually demanded by the clients. On the other hand, due to the client's accessing of the entire range of the data items, the overall skewness for Network N1 will be less and depend only on the value of the access skew coefficient θ.

Figs. 4–9 contains results for a different combination of server broadcasting speed and network environments for values of θ in [0 ... 1.5]. In each of these figures, six plots are presented. The plots termed "Non-adaptive" refer to the

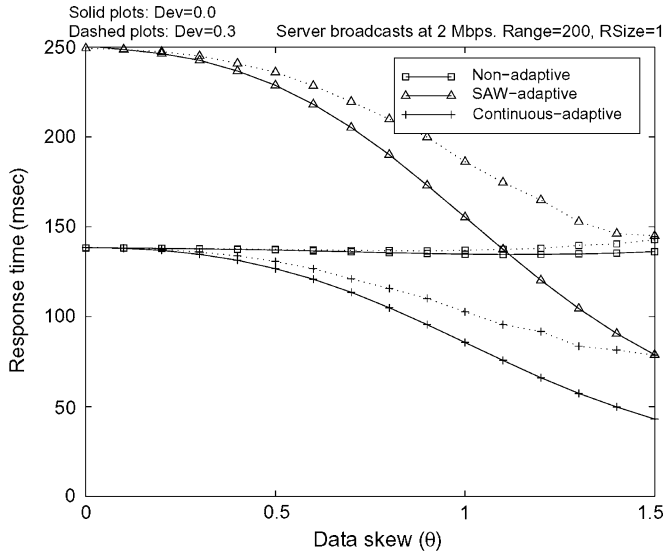

Fig. 4. Overall mean access time versus access skew coefficient θ for Network N1 and a 2 Mbps server broadcast speed.

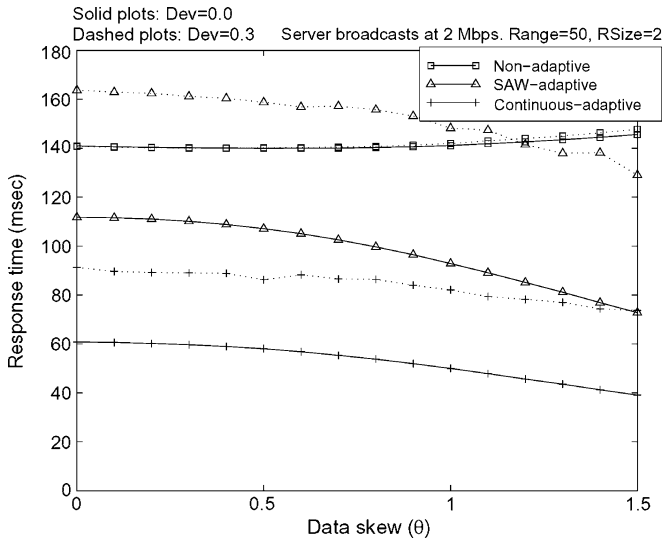

Fig. 5. Overall mean access time versus access skew coefficient θ for Network N2 and a 2 Mbps server broadcast speed.

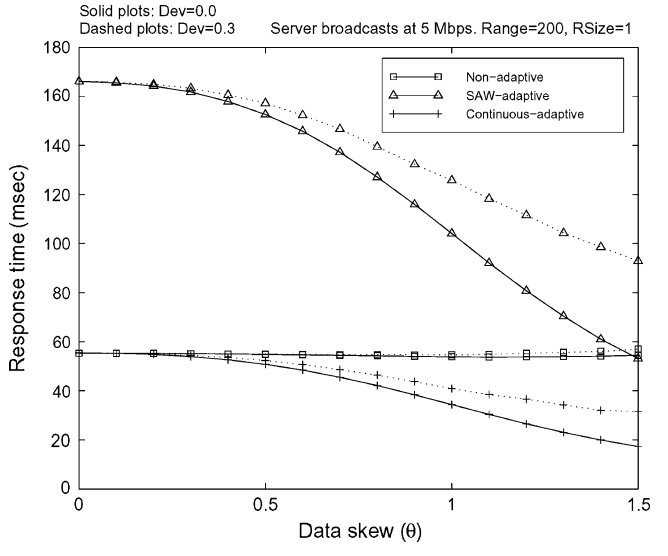

Fig. 6. Overall mean access time versus access skew coefficient θ for Network N1 and a 5 Mbps server broadcast speed.

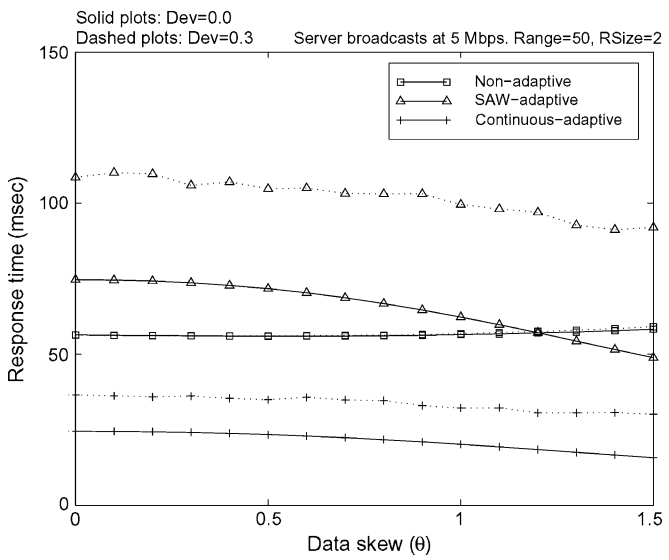

Fig. 7. Overall mean access time versus access skew coefficient θ for Network N2 and a 5 Mbps server broadcast speed.

non-adaptive broadcast of the server items [12] (which in this case are considered equiprobably demanded), the plots termed "SAW-adaptive" (Stop and Wait adaptive) refer to the method of [7] and the plots named "Continuous-adaptive" refer to the proposed method. The solid plots correspond to the performances of the systems with $Dev = 0$, whereas the dashed ones correspond to the performances of the systems with $Dev = 0.3$. Finally, Figs. 10 and 11 plot the performances of Networks N1 and N2 respectively, both for $\theta = 1.5$ and $Dev = 0.0$, under server broadcasting speeds that vary from 2 to 20 Mbps.

Figs. 4 and 5 depict the relative performances of the three systems for values of Dev equal to zero (solid plots) and 0.3 (dashed plots) for Networks N1 and N2 respectively for a 2 Mbps server broadcasting speed. In both of these figures the performance of the proposed system is always better than that of the non-adaptive and SAW-adaptive ones. This is because it efficiently utilizes high-speed broadcasting servers, due to the fact that the proposed method for collecting client feedback does not impose an overhead to item broadcasting. The method of [7] on the other hand, cannot efficiently utilize high-speed broadcasting servers as its performance is significantly lower (i.e. overall mean access time over the client population, also known as response time, is higher) than that of the other schemes. This is due to the fact that the fixed time that the server needs to wait to collect feedbacks after each item broadcast turns to be a performance bottleneck due to the very small duration of each item transmission time under high broadcasting speeds. Finally, the performance of the non-adaptive method of [12] does not improve with increasing values of θ due to the lack of a learning procedure for the items' demand probabilities.

Another observation from both of these figures is that the performance for the two adaptive systems is worse when $Dev =$

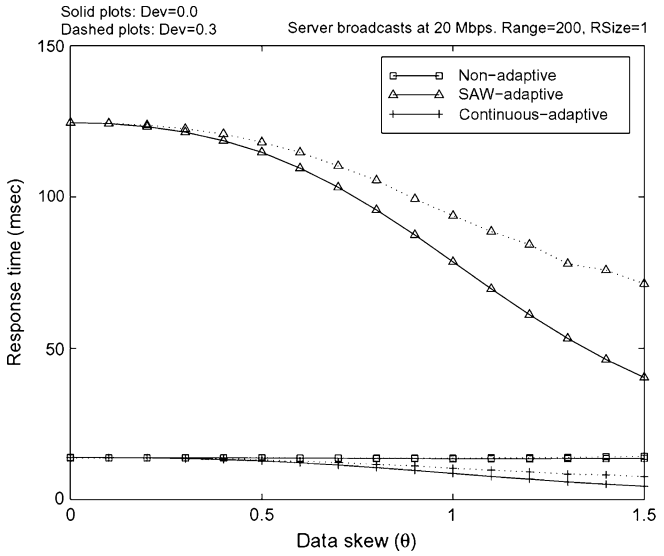

Fig. 8. Overall mean access time versus access skew coefficient θ for Network N1 and a 20 Mbps server broadcast speed.

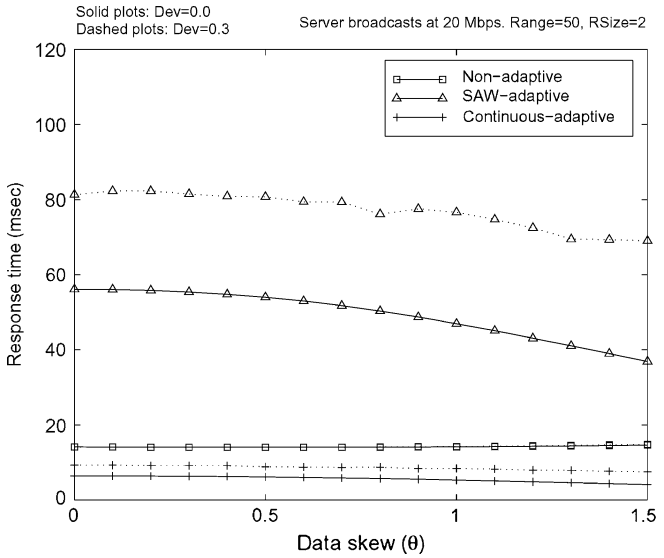

Fig. 9. Overall mean access time versus access skew coefficient θ for Network N2 and a 20 Mbps server broadcast speed.

0.3 compared to the cases of $Dev = 0$. This is expected behavior due to the decreasing commonality in client demands for increasing Dev. The non-adaptive system of [12] is not affected by an increasing Dev as in this system no learning of the demand probabilities takes place.

One can also see a different performance for the proposed system and the SAW-adaptive one of [7] when comparing Figs. 4 and 5. The performance of each system in N2 shows improved in Fig. 5 compared to that for N1 in Fig. 4, due to the fact that the use of $Range = 50$ for a database size of 200 that is used results to an increased skewness when compared with the skewness of N1 for the same values of θ. The increased skewness in turn translates to a better performance [7], [8], [12]. Moreover, the differences in the way the two systems perform in N1 and N2 can be seen from Figs. 4 and 5. These differences are also attributed to the fact that in N2, the use of $Range = 50$

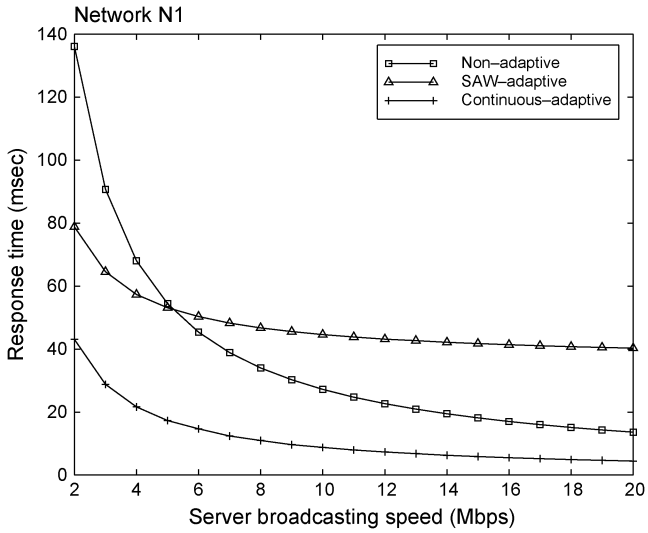

Fig. 10. Overall mean access time versus server broadcasting speed for Network N1. $Dev = 0.0$, $\theta = 1.5$.

leads the clients to select items only from a certain database part, a fact that yields increased skewness for N2 and therefore performance gains against N1 even for small values of θ [7].

Another interesting observation is that for small values of θ the performance of the proposed approach in Network N1 is close to that of the non-adaptive scheme. This is due to the facts that a) the non-adaptive system does not utilize client feedback and is thus not impaired with the aforementioned performance bottleneck of [7], b) in the proposed system, small values of θ tend to reduce the demand pattern to a uniform distribution over the server's data items, which is obviously a condition without demand skewness. For increasing values of θ however, the performance of the proposed system increases significantly compared to that of the non-adaptive one.

A final comment regarding Figs. 4 and 5 is that for Network N2, contrary to N1, the non-adaptive scheme of [12] is well out-performed by the proposed method even in the case of small values of θ, since it notifies the server to broadcast only the hot-spot of the database more frequently. Thus it would be useful even in cases of applications that access subsets of the server's database with demand patterns close to being uniform (as is the case of $\theta = 0$ for Network N2). For the SAW-adaptive method of [7] however, this advantage exists only for $Dev = 0$ and disappears when a higher than 2 Mbps broadcasting speed is used, as can be observed from the rest of the figures.

The pairs of Figs. 6–9 present results for the cases of using a 5 Mbps and a 20 Mbps broadcasting server respectively. These figures also exhibit the characteristics described above for Figs. 4 and 5. Additionally, one can observe that the performance gains of the proposed system against the SAW-adaptive one of [7] increase for a higher server broadcasting speed. Moreover, as seen from Figs. 10 and 11 that plot for various broadcasting speeds the performances of the three systems in N1 and N2 respectively, in high skewness environments, the performance difference remains substantial and is at least two times better for the proposed system when compared to the SAW-adaptive one.

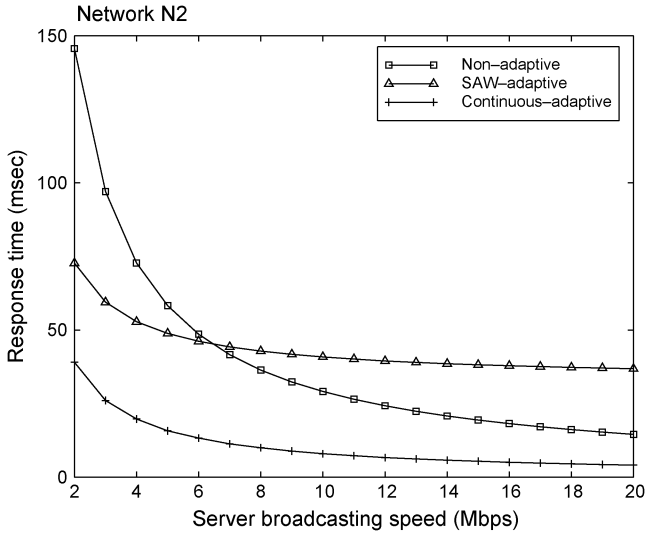

Fig. 11. Overall mean access time versus server broadcasting speed for Network N2. $Dev = 0.0, \theta = 1.5$.

Thus, the main conclusions that can be drawn form Figs. 4–11 are the following:

- The proposed approach efficiently utilizes high-speed broadcasting servers, contrary to the method of [7] whose performance does not scale with the server's broadcasting speed.
- For both the proposed approach and that of [7], performance decreases for increasing values of Dev.
- For small values of θ the performance of the proposed approach in Network N1 is close to that of the non-adaptive scheme and increases with an increasing skewness in item demands, as also noted in [7], [12].
- The proposed system is useful even in cases of applications that access subsets of the server's database with demand patterns close to being uniform (as is the case of $\theta = 0$ for Network N2).

V. Conclusion

This paper proposed an adaptive push system that operates efficiently in environments characterized by high server broadcasting speeds and a-priori unknown client demands for data items. The proposed system adapts to the demand pattern of the client population in order to reflect the overall popularity of each data item. We proposed a method for feedback collection by the server so that the client population can enjoy a performance increase in proportion to the broadcasting speed used by the server. Simulation results are presented which reveal satisfactory performance in environments with a-priori unknown client demands and under various high server broadcasting speeds.

Acknowledgment

The authors would like to thank the Associate Editor and the anonymous reviewers for their efforts in the review of this paper.

References

[1] S. H. Kang, S. Choi, S. J. Choi, G. Lee, J. Lew, and J. Lee, "Scheduling data broadcast based on multi-frequency in mobile interactive broadcasting," *IEEE Trans. Broadcasting*, vol. 53, no. 1, pp. 405–411, Mar. 2007.

[2] T. Yoshihisa, M. Tsukamoto, and S. Nishio, "A scheduling scheme for continuous media data broadcasting with a single channel," *IEEE Trans. Broadcasting*, vol. 52, no. 1, pp. 1–10, Mar. 2006.

[3] T. Yoshihisa, M. Tsukamoto, and S. Nishio, "A scheduling protocol for continuous media data broadcasting with large-scale data segmentation," *IEEE Trans. Broadcasting*, vol. 53, no. 4, pp. 780–788, Dec. 2007.

[4] L-S. Juhn and L-M. Tseng, "Adaptive fast data broadcasting scheme for video-on-demand service," *IEEE Trans. Broadcasting*, vol. 44, no. 2, pp. 182–185, Jun. 1998.

[5] L-S. Juhn and L-M. Tseng, "Fast data broadcasting and receiving scheme for popular video service," *IEEE Trans. Broadcasting*, vol. 44, no. 1, pp. 100–105, Mar. 1998.

[6] P. Nicopolitidis, M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis, *Wireless Networks*. New York: John Wiley and Sons, 2003.

[7] P. Nicopolitidis, G. I. Papadimitriou, and A. S. Pomportsis, "Using learning automata for adaptive push-based data broadcasting in asymmetric wireless environments," *IEEE Trans. Vehicular Technology*, vol. 51, no. 6, pp. 1652–1660, Nov. 2002.

[8] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-based data delivery using broadcast disks," *IEEE Personal Communications*, vol. 2, no. 6, pp. 50–60, Dec. 1995.

[9] S. Acharya, M. Franklin, and S. Zdonik, "Prefetching from a broadcast disk," in *Proceedings of the 12th International Conference on Data Engineering*, New Orleans, U.S.A, Mar. 1996, pp. 276–285.

[10] S. Acharya, M. Franklin, and S. Zdonik, "Disseminating updates on broadcast disks," in *Proceedings of the 2nd VLDB Conference*, Bombay, India, Sep. 1996, pp. 354–365.

[11] S. Acharya, M. Franklin, and S. Zdonik, "Balancing push and pull for data broadcast," in *Proceedings of ACM SIGMOD*, 1997, pp. 183–194.

[12] N. H. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments," *Wireless Networks*, vol. 5, no. 3, pp. 171–182.

[13] C. J. Su, L. Tassiulas, and V. J. Tsotras, "Broadcast scheduling for information distribution," *Wireless Networks*, vol. 5, no. 2, pp. 137–147, Mar. 1999.

[14] N. Vaidya and S. Jiang, "Response time in data broadcast systems: Mean, variance and trade-off," in *Proceedings of WOSBIS*.

[15] N. Vaidya and S. Jiang, "Scheduling data broadcast to impatient users," in *Proceedings of International Workshop on Data Engineering for Wireless and Mobile Access*, pp. 52–59.

[16] P. Nicopolitidis, G. I. Papadimitriou, and A. S. Pomportsis, "Exploiting locality of demand to improve the performance of wireless data broadcasting," *IEEE Trans. Vehicular Technology*, vol. 55, no. 4, pp. 1347–1361, Jul. 2006.

[17] P. Nicopolitidis, G. I. Papadimitriou, and A. S. Pomportsis, "Multiple antenna data broadcasting for environments with locality of demand," *IEEE Trans. Vehicular Technology*, vol. 56, no. 5, pp. 2807–2816, Sep. 2007.

[18] V. Kakali, G. I. Papadimitriou, P. Nicopolitidis, and A. S. Pomportsis, "A new class of wireless push systems," *IEEE Trans. Vehicular Technology*, 2008, submitted for publication.

[19] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Upper Saddle River, NJ: Prentice Hall, 1989.

[20] G. I. Papadimitriou and A. S. Pomportsis, "Learning automata-based TDMA protocols for broadcast communication systems with bursty traffic," *IEEE Communication Letters*, vol. 4, no. 3, pp. 107–109, Mar. 2000.

[21] G. I. Papadimitriou and A. S. Pomportsis, "Self-adaptive TDMA protocols for WDM star networks: A learning-automata-based approach," *IEEE Photonics Technology Letters*, vol. 11, no. 10, pp. 1322–1324, Oct. 1999.

[22] G. I. Papadimitriou and D. G. Maritsas, "Learning automata-based receiver conflict avoidance algorithms for WDM broadcast-and-select star networks," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 407–412, Jun. 1996.

[23] P. Nicopolitidis, G. I. Papadimitriou, and A. S. Pomportsis, "Learning-automata-based polling protocols for wireless LANs," *IEEE Trans. Communications*, vol. 51, no. 3, pp. 453–463, March 2003.

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

[25] R. Jain and J. Werth, *Airdisks and airRAID: Modeling and Scheduling Periodic Wireless Data Broadcast (extended abstract)*. Piscataway, NJ: Rutgers-The State University, 1995.

[26] K. S. Gilhousen, I. M. Jacobs, R. Padovani, A. J. Viterbi, L. A. Weaver, J. , and C. E. Wheatley, "On the capacity of a Cellular CDMA System," *IEEE Trans. Vehicular Technology*, vol. 40, no. 2, May 1991.
[27] C. Y. Lee, "Overview of cellular CDMA," *IEEE Trans. Vehicular Technology*, vol. 40, no. 2, May 1991.

Petros Nicopolitidis received B.S. and Ph.D. degrees in computer science from the Department of Informatics, Aristotle University of Thessaloniki, Greece, in 1998 and 2002, respectively. Since 2004 he is a Lecturer in the same department. He has published more than 50 papers in international refereed journals and conferences. He is co-author of the book Wireless Networks (Wiley, 2003). His research interests are in the areas of wireless networks and mobile communications. Since 2007 he serves as an associate editor for the International Journal of Communication Systems published by Wiley. He is a member of IEEE.

Georgios I. Papadimitriou (SM'90) 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 and 2001 to 2006, he was a Lecturer and Assistant Professor respectively at the Department of Informatics, Aristotle University of Thessaloniki, Greece. Since 2006 he is an Assiociate Professor at the same Department. His research interests include optical networks, wireless networks, high speed LANs and learning automata. Prof. Papadimitriou is Associate Editor of five scholarly journals, including the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS—PART C, the IEEE TRANSACTIONS ON BROADCASTING, and the IEEE Communications Magazine. He is co-author of the books "Multiwavelength Optical LANs" (Wiley, 2003) and "Wireless Networks" (Wiley, 2003) and co-editor of the book "Applied System Simulation" (Kluwer, 2003). He is the author of more than 150 refereed journal and conference papers. He is a Senior Member of IEEE.

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.