A New Class of Wireless Push Systems

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

***Abstract*—Data broadcasting is an efficient way of delivering information over asymmetric wireless environments, and push systems can provide high scalability and client hardware simplicity. Many environments, however, are characterized by *a priori* unknown client demands and groups of clients that are located at the same region and have similar demands. The main disadvantage of the approaches in such cases is the lack of fairness, because groups with few members have much lower performance than groups with many clients, because the performance per group (directly) depends on the group size. In this paper, we propose a fair push system where the performance of each client is independent of the total number of clients that are located in the same region. This condition is achieved without the overall performance of the system being significantly affected. Moreover, the proposed fair system is extended to a priority-based system that multiplies the performance of a specific group, depending exclusively on its priority level in relevance to the rest of the groups.**

***Index Terms*—Broadcasting, fairness, group size, locality, priorities, push system.**

I. Introduction

DATA broadcasting is an efficient way of delivering information over asymmetric wireless networks (i.e., traffic information, weather information, and news distribution). In such applications, clients' preferences are usually overlapping, and the broadcast of a single information item will likely satisfy a large number of clients. There are three major approaches for designing broadcast schedules. The pull systems (for example, see [1]), where the server schedules its broadcasts using the clients' requests, are adaptive to dynamic client demands but are not scalable for large client populations, because client requests will either collide with each other or saturate the server. In push systems (for example, see [2]–[4]), the server makes broadcasts according to an *a priori* estimate of the demand per information item. Push systems provide high scalability and client hardware simplicity. However, the pure-push systems cannot efficiently operate in environments with *a priori* unknown and dynamic client demands. The adaptive push system in [5] and [6], however, achieves efficient operation in such environments by using a learning automaton at the broadcast server to provide adaptivity. After each item broadcast, every satisfied client sends a feedback to the server. Using the clients' feedbacks, the server's automaton continuously adapts to the overall client population demands to reflect the overall popularity of each data item. Finally, hybrid systems (for example, see [7]) try to combine the benefits of the pure-push and pure-pull approaches. In this paper, we will focus on the adaptive push-based approach in [5].

In data broadcasting, many environments are characterized by locality of demand [8]. Locality of demand means that clients belong to groups, with each one located at different regions, and members of each group have similar demands for information items, which are different from the demands of clients in other groups. For example, clients that are located around an airport have different information demands from the clients around a theater. In such an environment (with locality of demand), the system in [5] does not ensure fairness among clients, because the response time of a client and the number of server's broadcasts that concern this client are affected by the total number of clients that are located in the same place. In particular, the large number of feedbacks from a large group will indicate the server to broadcast much frequently data items that concern this group, at the expense of the smaller groups. Thus, the server will more frequently broadcast data items to groups with many clients than to groups with few members.

Fairness is an important issue in data broadcasting and appears in different aspects in the literature. The approach in [9] concerns a pull system that provides fairness in the case of items of different sizes at the broadcast server, whereas the approach in [10] concerns an adaptive balanced scheme that aims at the fairness between the access times of the "pushed" and "pulled" items for a hybrid data delivery in a multichannel data dissemination environment. The approach in [11], using the response time fairness as a metric, studies a data broadcast system, taking into account the broadcast and disk scheduling, as well as the cache management, of the system. The aspect of fairness that is considered in this paper concerns the lack of fairness that is observed due to the unequal size of the clients' groups. To our knowledge, this paper is the first approach that deals with this aspect of fairness.

In this paper, we propose a push-based system that achieves both adaptivity and fairness in an environment with locality of demand, unequal group sizes, and dynamic client demands, without significantly affecting the overall performance of the system. Moreover, a priority mechanism has been introduced to the proposed push-based system, which allows different priority levels among the geographic locations where the provider offers his/her services. This mechanism achieves the multiplication of the performance of a specific group of clients, depending exclusively on its priority level in relevance to the rest of the groups and not on the popularity of the specific group. The remainder of this paper is organized as follows. Section II comprises

Manuscript received September 24, 2007; revised September 2, 2008, December 10, 2008, and March 16, 2009. First published May 19, 2009; current version published October 2, 2009. The review of this paper was coordinated by Prof. Y.-B. Lin.

The authors are with the Department of Informatics, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece (e-mail: vicky198@csd.auth.gr; gp@csd.auth.gr; petros@csd.auth.gr; apompo@csd.auth.gr).

Digital Object Identifier 10.1109/TVT.2009.2023256

a brief introduction to learning automata and a description of the system architecture and presents the proposed fair push-based approach. Section III presents the proposed priority mechanism. The simulation results are presented in Section IV, and Section V concludes this paper.

II. Fair Push-Based System

A. Learning Automata

Learning automata [12] are mechanisms that 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 A actions the optimal one so that the average penalty received by the environment is minimized. This condition 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 the 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 $b(n)$ that was triggered by the action a_i selected at cycle n to update the probability distribution vector p. After the update is finished, the automaton selects the action that will be performed at time cycle $n+1$ according to the updated probability distribution vector $p(n+1)$. A general reinforcement scheme has the form of (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 $b(n)$, and updates the probability distribution vector p. $a(n)$ is the action that was executed in cycle n, and $b(n)$ is a metric of the environmental response, which is normalized in [0,1]. The lower the value of $b(n)$, the more favorable the response. When $b(n)$ takes continuous values after normalization in the interval [0,1], the automaton is known as an S-model [13]. The functions g_i and h_i are associated with reward and penalty for action a_i, respectively. The g_i contributes a term that is proportional to $[1 - b(n)]$, and h_i contributes a term that is proportional to $b(n)$ [12]. We have

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

In data networking, learning automata have been applied to several problems, including the design of self-adaptive medium access control protocols for wired and wireless platforms, which efficiently operate in dynamic workloads (for example, see [14]).

B. System Architecture

The system consists of a base station and mobile clients. The base station consists of one omnidirectional antenna and a server that contains a database of items and is equipped with a learning automaton. Clients are considered equipped with Global Positioning System (GPS) receivers. Currently, many personal digital assistants (PDAs) are typically equipped with GPS receivers; thus, there is no need for complex client equipment. In [15], the extended use of GPS receivers in mobile phones and PDAs is revealed. As mentioned in [15], Forrester Research estimated that in 2006, there were more than 100 million subscribers in the U.S. with GPS in their phones, and about 65 million more would be sold in 2007. This extended use of GPS has inevitably led to the development of many location-based services (LBSs). In the literature, many LBSs have been proposed using mobile phones or PDAs that were equipped with GPS receivers [16]–[19]. Moreover, many of the proposed LBSs have already commercially been applied, including applications like child tracking and tracking the whereabouts of friends (BuddyFinder). In [15], it is mentioned that, in its September 2006 U.S. Wireless Business Location-Based Services 2006–2010 Forecast Report, IDC predicted that more than half of U.S. mobile-phone users will begin using LBSs within four years.

The base station broadcasts the server's data items, whereas clients respond ("vote") to the server's broadcasts. For the uplink communication, a code-division multiple access (CDMA) coding has been chosen. After the broadcast of an item that has been expected from a client, the client's software application automatically sends its "vote" (i.e., 1 bit) to the base station using a user-specific high-speed code (long code). One issue that has to be considered is how long the server has to wait until he receives the clients' votes. The voting is automatically performed by the client's software according to the user's preferences that were recorded by the client's software at earlier times, and the client is not directly involved in this process; thus, the server shall simply wait for a time period that is at least equal to the sum of twice the propagation delay to the more distant client (this delay is determined by the server's coverage area), plus the item transmission time by the server, plus the feedback transmission time by the client. At the receiver end (base station), the signals are separated using a correlator (rake receiver), which only accepts signals energy from the specific client's long code and despreads its spectrum. Other couser signals remain spread, because their spreading algorithm is uncorrelated with the desired signal's algorithm, and they appear as noise. Moreover, the server does not take into consideration more than one vote from a CDMA code that has already been used to avoid the case where the application of the same client has voted faulty more than once.

The CDMA technique has been preferred, because CDMA has advantages in several aspects compared with other multiple access techniques [20], [21], which are listed as follows:

1) *Multiple forms of diversity (time, frequency, and space)*. This factor effectively reduces the impact of multipath fading.
2) *High capacity*. The primary purpose of using CDMA is to achieve high capacity. CDMA operates on a common

frequency carrier that implies a nearly one-in-one frequency reuse pattern. On the other hand, CDMA can tolerate noise to a great extent due to its wide bandwidth. All these factors contribute to the high capacity of CDMA systems [20], which is given by $N = ((W/R)/(E_b/N_o)) + 1$, where N are the clients of the system, W represents the system transmission bandwidth, R represents the transmission bit rate, and E_b/N_o is the bit energy-to-noise power spectral density. Suggestively, for $W = 5$ MHz, $R = 1024$ b/s, and $E_b/N_o = 7$ dB $= 5$, a population of $N = 977$ clients is calculated. In case that there is a need for more clients, the implementation of CDMA in more than one frequency channel can increase the required capacity.

3) *Noise suppression*. The CDMA receiver can operate in a very low signal-to-noise ratio (SNR) environment due to its high processing gain.
4) *Low RF transmit power*. The CDMA transmitters always transmit at the lowest possible power level. This approach helps minimize interference and save mobile battery life.

Note that 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 1 bit coded with its long code); thus, a simple implementation of a CDMA coding is considered more than enough to guarantee correct reception of data that were sent by the clients. Moreover, although CDMA has the potential to deal with fading due to multipath effects (i.e., diversity), 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, because multiple signals that are diffracted by various obstacles add up with different propagation delays. The maximum allowed bit rate is given by $R\max = 1/(2 \cdot DS)$, where DS is the delay spread that characterizes multipath in a specific propagation environment. Typical values of the delay spread are 0.2 μs for propagation in an irregular terrain, 0.5 μs for a suburban environment, and 3 μs for an urban environment. With the worst value being 5 μs, the maximum bit rate is 100 Kb/s; hence, the choice of a few kilobits per second is enough to guarantee full suppression of multipath effects.

C. Broadcasting Algorithm

To optimize performance, broadcast schedules must be periodic [22], and the variance of spacing between consecutive instances of the same item must be reduced [23]. Based on the aforementioned conditions, the broadcast scheduling of many push systems [3], [5] are based on the following arguments, which are proven in [3].

1) Schedules with the 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 the 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 that an item of length l_i is received with an unrecoverable error.

In [3], a nonadaptive push system is presented, and it is revealed that a broadcast algorithm based on the aforementioned arguments minimize the mean response time of the system (optimize performance). In an environment where the client demand probabilities p of all the server's items are equal to each other and all the items have the same length l, the system in [3] degenerates into a simple round-robin broadcast of the items (flat broadcasting). Our proposed push system is adaptive and fair. The adaptivity mechanism is the same as the one used in [5]. Thus, the simulation results in [5] show off the adaptivity of the system, whereas the fairness will be presented in Sections II-E and F. The broadcasting algorithm that is used in this paper is based on the aforementioned arguments, is the same as the one used in [5], and operates as follows.

The broadcast server is equipped with a learning automaton that contains the server's estimate p'_i of the demand probability p_i for each data item i that the server broadcasts. Obviously, $\sum_{i=1}^{M} p'_i(n) = 1$, where M is the number of the data items. For example, in a system that broadcasts traffic information, the server's database contains 200 items of traffic information about the area that is covered by the server's antenna.

Assuming that T is the current time and $R(i)$ is the time when item i was last broadcast, for each broadcast, the server selects to transmit the item i that maximizes the cost function (objective function), i.e.,

$$CF(i) = (T - R(i))^2 \, (p'_i/l_i) \, (1 + E(l_i)) \, / \, (1 - E(l_i)) \qquad 1 \le i \le M \quad (2)$$

where p'_i is the demand probability for item i, l_i is the item's length, $E(l_i)$ is the probability that an item of length l_i is received with an unrecoverable error, $R(i)$ is initialized to -1, and if the maximum value of $CF(i)$ is given by more than one item, the algorithm arbitrarily selects one of them. Upon the broadcast of item i at time T, $R(i)$ is changed so that $R(i) = T$.

After the transmission of item i, the server waits for an acknowledgement from every client that was waiting for item i. This approach means that each client who is satisfied with the server's last broadcast transmits a feedback to the server.

To achieve fairness to the system because of the unequal group sizes, the server normalizes the clients' feedbacks depending on the population of the group to which they belong, as will be described in Section II-F.

After the reception of the clients' feedbacks, the server updates the demand probability vector p' of the learning automaton. For the next broadcast, the server chooses which item i, $1 \le i \le M$, will be transmitted using the updated vector p' and the cost function (2).

D. Probability-Updating Scheme

When there are no satisfied clients after the transmission of item i, the probability estimates (vector p') do not change. Due to the nature of a wireless communication that is subject to errors, the probability of the item i does not decrease when

there are no satisfied clients so that the possibility of an item i to be considered unpopular (zero votes) due to a faulty (or no) reception of the broadcast item by the whole number of clients is avoided.

However, when there are satisfied clients, the probability of the transmitted item i increases whereas the probabilities of all the rest of the items decrease. The learning automata scheme (3) in this paper derives from the one in (1) for penalty function h_i that is equal to zero according to [12]. Equation (3) is the same as the one that has widely been used in many other protocols for wired and wireless platforms that use the learning automata [5], [6], [8], [13], [14]. Considering that the server's kth transmission is item i, the probability vector p' is updated as follows:

$$\begin{aligned} p'_j(k+1) &= p'_j(k) - L\,(1-b(k))\left(p'_j(k)-a\right) \qquad \forall j \neq i \\ p'_i(k+1) &= p'_i(k) + L\,(1-b(k))\sum_{i\neq j}\left(p'_j(k)-a\right). \end{aligned} \tag{3}$$

Equation (3) stems from (1) by setting $g_i(p(n)) = L(p'(n) - a)$ and $h_i(p(n)) = 0$, where L is a parameter that governs the speed of the automaton convergence. The lower the value of L, the more accurate the estimation made by the automaton—a fact, however, that comes at the expense of the 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 items. If the probability estimate p'_i of an item i become zero, then $CF(i)$ would be very close to zero. However, the item, even if it is unpopular, still needs to be transmitted, because some clients may request it. The role of parameter α is to prevent the probabilities of unpopular items from taking values in the neighborhood of zero to increase the adaptivity of the automaton.

Upon reception of the clients' feedbacks, this number of feedbacks is normalized in the interval of [0,1]. $b(k) = 1 -$ (the number of received feedbacks/the total number of clients) is the system environmental response that is triggered after the server's kth transmission. The server possesses an estimate of the total number of clients that are under its coverage in a way that will be described in the following discussion. A value of $b(k)$ that is equal to one represents the case where no client feedback is received. Thus, the lower the value of $b(k)$ is, the more that clients were satisfied by the server's kth transmission.

For the next broadcast, the server chooses which item will be transmitted using the updated vector p'.

E. Significance of Fairness in a Push-Based System With Locality of Demand

To indicate the significance of fairness in a system with unequal sizes of the clients' groups, we present the following situation. We consider that Group A has much fewer members than Group B and that the server has broadcast a data item that is demanded by Group A. Due to the few members of Group A, the server will receive a small number of feedbacks. Thus, the server will consider that this item is unpopular at the system and will not broadcast it in the near future, although this item has satisfied the majority of clients at Group A. On the contrary, the large number of feedbacks from the large Group B indicates the server to much frequently broadcast data items that concern Group B. This way, the percentage of the server's broadcasts that concern Group A is much less than the percentage of the server's broadcasts that concern Group B. Thus, the mean response time of each client at Group A is much higher (much lower performance) than the mean response time of each client at Group B. To indicate this condition, the following situation is presented. Two groups, i.e., Groups A and B, are considered. The vector Size denotes the number of clients at each group, and it holds that $\text{Size}(B) = 2\text{Size}(A)$. The vector W denotes the weight of the vote of each group, and because the vote of each client is equally weighted, it holds that $W(A) = W(B)$. MaxFeedback is defined as the maximum feedback that the server receives in the case where the broadcast item satisfies the entire group, i.e., $\text{MaxFeedback} = \text{Size} \cdot W$. In the aforementioned situation, it holds that $\text{MaxFeedback}(A) = \text{Size}(A) \cdot W(A)$ and $\text{MaxFeedback}(B) = \text{Size}(B) \cdot W(B) = 2\text{Size}(A) \cdot W(A) = 2 \cdot \text{MaxFeedback}(A)$. Thus, the MaxFeedback of Group B is double the MaxFeedback of Group A because of the double size of Group B. In other words, the sum of the feedbacks, which defines the item's popularity [term $b(k)$ in (3)], depends on the groups' size. The lack of fairness is obvious, because clients with the "bad luck" of belonging to a small group are rarely served, whereas clients of large groups are served at regular times.

The aforementioned situation is an undesirable one in a data-broadcasting system. In such a system, each client, as a subscriber to the service provider, demands to be served as fairly as the rest of the subscribers without the services that are provided to him being dependent on the existence or absence of several clients who happen to simultaneously have the same items' demands.

F. Proposed Feedback Mechanism Scheme to Provide Fairness

The fairness of the system lies in the independence of the items' popularity and, thus, of the items' broadcast frequency from the groups' size. In other words, the sum of the feedbacks, which defines the item's popularity [term $b(k)$ in (3)], must become independent of the groups' size. This paper proposes the use of different weights to the clients' feedbacks based on the size of the group to which they belong. In other words, the server will normalize each feedback based on its origin. The proposed feedback mechanism scheme that provides fairness is described as follows

$$\begin{aligned} \text{MaxFeedback}(1) &= \text{MaxFeedback}(2) \\ &= \cdots = \text{MaxFeedback}(m) \qquad (4.a) \\ W(1)\text{Size}(1) &= W(2)\text{Size}(2) = \cdots = W(m)\text{Size}(m) \qquad (4.b) \end{aligned}$$

where m is the number of groups, $\text{MaxFeedback}(1)$, $\text{MaxFeedback}(2), \ldots, \text{MaxFeedback}(m)$ is the maximum feedback of each group, $\text{Size}(1), \text{Size}(2), \ldots, \text{Size}(m)$ is the number of clients at each group, and $W(1), W(2), \ldots, W(m)$ refers to the different weights that the server sets to the feedbacks of each group.

Lemma 1: When the ratio of the sizes of two groups is equal to k, $(\text{Size}(A)/\text{Size}(B)) = k$, then the ratio of the respective feedback's weights is equal to $1/k$, $W(A)/W(B) = 1/k$.

Hence, there must be a mechanism that will define the different weights that the server will set on the clients' feedbacks. This mechanism works as follows. The server, which has *a priori* knowledge of the geographic locations (coordinates) where his services are offered, consecutively sends one control packet for each of these locations, asking of the clients that belong to this location to send back a feedback. The client's GPS receiver detects its coordinates, and the client sends its feedback (vote) as an answer to the control packet that refers to its area. Then, the server, using its long-code database, decodes the votes that are the answers to each control packet and builds the groups based on the decoded votes that were received from each area.

Privacy is an issue of consideration in the LBS, where knowledge of the location of an individual or an object is used to personalize the service. One typical example of LBS is the commercial BuddyFinder, where the provider, based on the geographical position of its client, informs the clients about the presence of their "friends" in the area. Location technology comes in the following two forms: 1) tracking and 2) positioning. Using the tracking technology, the position is computed by some external entity (e.g., network operator). The term positioning is often used when referring to the technology that allows the located object to compute its location by itself (e.g., GPS). In case that the positioning technology is used in conjunction with a local mapping software, privacy may not be an issue, which is in contrast to location services based on the tracking technology, where the provider of the LBS is different from the location provider [24]. In the proposed scheme, the answer that each client sends to the server's control packet that refers to his/her area is a simple "vote" (i.e., 1 bit) similar to the one that sends as a feedback to the items' broadcasts. This "vote" is anonymous and does not contain any kind of client's personal information, e.g., phone number or name. Thus, no issue of user's privacy is raised for the proposed system of positioning technology or, at least, no further issue beyond the known privacy issues of mobile communication.

Through the above approach, the server, using the number of the decoded feedback votes that he/she receives as an answer to the control packet that refers to each area, builds the groups and obtains an estimation of the size of each group. Then, the server, by taking into account the size of each group and (4.b), calculates $W(1), W(2), \ldots, W(m)$. Moreover, the group sizes may dynamically change; thus, the server periodically repeats the aforementioned procedure to calculate the new values of vector W. Hence, in the proposed fair scheme, upon the server's broadcast of one item that refers to group i at time k and the reception of the clients' feedbacks (votes), term $b(k)$ is defined as $b(k) = 1 - (W(i) \cdot$ the number of received feedbacks$/W(i) \cdot \text{Size}(i))$, and this ratio is used in the update procedure of the learning automaton and, thus, in the broadcast scheduling procedure.

In cases of indoor location services, some clients may find difficulties in updating their GPS data. In such cases, each client sends its feedback as an answer to the server's control packet that refers to his/her latest stored GPS data, which are the coordinates of the nearest outdoor area. For example, in case of a theater, the latest stored GPS data are the coordinates of the area outside the theater. The possibility of two or more location services lying in adjacent areas is particularly low; thus, those clients are considered to be members of the indoor location service in which they lie. However, in a borderline case where two or more location services lie in adjacent areas and a client does not respond to the server's control packet that refers to his/her real coordinates due to the nonupdated GPS data, the client is wrongly considered by the server to be a member of a group to which it does not belong. However, this borderline case has no significant effect on the performance of the proposed system, as revealed by the simulation results in Section IV.

III. Priorities-Based Scheme

A. Using Priorities in a Push-Based System With Locality of Demand

The aforementioned proposed system does not support priorities, because it considers a system where all the locations that the server provides his/her services have the same priority level, and thus, equal performance is achieved among groups.

However, there are groups where the high frequency of items' broadcasts is critical (e.g., traffic condition) and are considered as groups of high priority level, which is contrary to low-priority groups (e.g., sights around an area). The higher the priority level of a group, the higher the frequency of the broadcasts that refers to this group. However, the broadcast frequency of an item i is determined by the frequency that the item is picked by the cost (objective) function, because this function is the one that chooses the item that will be transmitted. Thus, items that refer to high-priority-level groups should more frequently be picked by the cost function than the ones that refer to low-priority groups. Therefore, the aforementioned scheme can be extended to a system that supports the existence of priority levels among groups. The priority mechanism can be implemented using different weights for the feedbacks ("votes") of each client based on the group to which he/she belongs. However, in environments of unequal group sizes, the calculation of a weight vector W that is related only to the priority level of each group is not sufficient to reflect the use of different priorities to the performance of the system. To indicate this case, we present the following situation. We consider Groups A and B, where $\text{Size}(B) = 2\text{Size}(A)$, and $\text{Priority}(A) = 2\text{Priority}(B)$. To double the performance of Group A, the maximum feedback that the server receives from Group A, which defines the item's popularity [term $b(k)$ in (3)], should be double the one from Group B. In other words, $\text{MaxFeedback}(A) = 2\text{MaxFeedback}(B)$. If we simply double the weight of Group A's feedbacks, i.e., $W(A) = 2W(B)$, then

$$\text{MaxFeedback}(A) = \text{Size}(A)W(A)$$

$$\text{MaxFeedback}(B) = \text{Size}(B)W(B)$$

$$= 2\text{Size}(A)\frac{W(A)}{2} = \text{Size}(A)W(A).$$

Thus, $\text{MaxFeedback}(B) = \text{MaxFeedback}(A)$, which means that despite the doubling of the weight of Group's A feedbacks, Groups A and B seem to be of the same priority level, because their unequal group sizes were not taken into consideration.

B. Proposed Feedback Mechanism to Provide Priorities to a Push-Based System

To provide such a system with priorities, the weight vector W must be calculated and related on both the priority level and group size of each group. Considering that m is the number of groups, $\text{MaxFeedback}(1), \text{MaxFeedback}(2), \ldots, \text{MaxFeedback}(m)$ is the maximum feedback of each group, $\text{Size}(1), \text{Size}(2), \ldots, \text{Size}(m)$ is the number of clients at each group, $A(1), A(2), \ldots, A(m)$ is the priority coefficient of each group, and $W(1), W(2), \ldots, W(m)$ is the weight for each group, the ratio of the maximum feedback of one group to the maximum feedback of another group must be equal to the ratio of the respective priority coefficients, which is described as follows:

$$\frac{\text{MaxFeedback}(2)}{\text{MaxFeedback}(1)} = \frac{A(2)}{A(1)}$$
$$\frac{\text{MaxFeedback}(3)}{\text{MaxFeedback}(1)} = \frac{A(3)}{A(1)}, \ldots, \frac{\text{MaxFeedback}(m)}{\text{MaxFeedback}(1)} = \frac{A(m)}{A(1)} \tag{5.a}$$

$$W(2) = \frac{A(2)}{A(1)} \frac{W(1)\text{Size}(1)}{\text{Size}(2)}$$
$$W(3) = \frac{A(3)}{A(1)} \frac{W(1)\text{Size}(1)}{\text{Size}(3)}$$
$$W(m) = \frac{A(m)}{A(1)} \frac{W(1)\text{Size}(1)}{\text{Size}(m)}. \tag{5.b}$$

Lemma 2: When the ratio of the sizes of two groups is equal to k, i.e., $\text{Size}(A)/\text{Size}(B) = k$, and the ratio of the respective priority coefficients is equal to n, i.e., $A(A)/A(B) = n$, then the ratio of the respective weights is equal to n/k, i.e., $W(A)/W(B) = n/k$.

Using the aforementioned priority mechanism, the server dynamically alters the priority levels among the groups, depending on the altering priority needs of the system. For example, in cases of an area with urgent traffic or weather conditions, the server significantly increases the priority level of these locations at the expense of other locations with nonurgent conditions (e.g., the sights of an area) to achieve higher performance at the locations with urgent needs.

IV. Performance Evaluation

Using simulation, we compared the proposed fair push system with the system in [5] with the nonweighted feedbacks in an environment with the following characteristics:

1) *a priori* unknown client demands to the server;
2) locality of demand;
3) unequal group sizes;
4) priorities.

To evaluate the fairness of the proposed fair system compared with the system in [5], we have chosen to use the "percentage of broadcasts per group" and the "mean response time per group" as the performance metrics. The mean response time per group is defined as the mean time units that the clients at each group wait until the item in which they are interested is broadcast. Obviously, as the values of the mean response time per group approach each other, the system becomes fairer. This result occurs, because the phenomenon of the starvation of some groups (very high response time) in favor of other groups whose demands are served at regular times (very low response time) is eliminated. The percentage of broadcasts per group is defined as the number of the server's broadcasts of items that concern each group upon the total number of the server's broadcasts. It is obvious that the same (or almost the same) values of server's broadcasts per group indicate that the server equally serves the clients that are under its services, independent of the size of the group. Finally, we have used the "overall mean response time of the system," which is defined as the time units that each client (independent of the group to which he/she belongs) waits at the average until the item in which he/she is interested is broadcast to reveal that the proposed approach does not significantly affect the overall performance of the system. The measure of the response time is considered to be the duration of the broadcast of one item. We have chosen to use the aforementioned metrics, which are different from the ones in [3], because our system is adaptive and fair compared with the nonadaptive unfair system in [3]. The simulation environment is described as follows.

A. Server Model

We consider a server that contains a database of size M in items. To avoid increasing the server's hardware complexity, we consider that the server has no knowledge of which items concern each location (group of clients). The item length l is considered to be the same for each item and equal to the unit. The server is initially unaware of the demand of each item; thus, initially, every item has a probability estimate p'_i of $1/M$. In the case of the environments that dynamically change their group sizes, the server sends a control packet every SrvCP broadcasts to calculate the new values of $\text{Size}(1), \text{Size}(2), \ldots, \text{Size}(m)$ and $W(1), W(2), \ldots, W(m)$.

B. Client Model

We consider a client population of NumCl. Clients are grouped into G groups, each one of which is located at a different location. To model groups with different group sizes, we compute the size of each group using the Zipf distribution. Thus, the ratio of the number of clients in group g, i.e., $1 \leq g \leq G$, to the number of clients in the entire system is

$$c\left(\frac{1}{g}\right)^{\theta}, \quad \text{where } c = 1 \Big/ \sum_{k}\left(\frac{1}{k}\right)^{\theta}, \quad k \in [1, \ldots, G] \tag{6}$$

where θ is a parameter named access skew coefficient. For $\theta = 0$, the Zipf distribution reduces to a uniform distribution of group sizes. For large values of θ, the Zipf distribution increasingly produces skewed patterns.

TABLE I
CHARACTERISTICS OF EACH SIMULATION ENVIRONMENT

Network	G	θ	M	$NumS_1$, $NumS_2$,..., $NumS_G$	θ_1	$A_1, A_2, ..., A_G$	Emigration
N_1	10	0.8	200	equal	0.5	equal	No
N_2	5	0.2	100	[23, 18, 14, 25, 20]	0.7	equal	No
N_3	5	0.4	100	[23, 18, 14, 25, 20]	0.7	equal	No
N_4	5	0.7	100	[23, 18, 14, 25, 20]	0.7	equal	No
N_5	5	0.9	100	[23, 18, 14, 25, 20]	0.7	equal	No
N_6	10	$\in\{0.1, ..., 0.9\}$	200	equal	0.4	equal	No
N_7	5	0.7	100	equal	0.4	[1, 2, 3, 4, 5]	No
N_8	5	0.7	100	equal	0.4	[1, 1.5, 1.5, 2, 3]	No
N_9	10	0.7	200	equal	0.5	equal	Yes
N_{10}	10	0.7	200	equal	0.5	[1, 1, 1.5, 1.5, 2, 2, 2.5, 2.5, 3, 3]	Yes

Any client that belongs to Group g is interested in the same subset S_g of server's data items. All items outside this subset have a zero-demand probability at the clients of the group. Moreover, $S_i \neq S_j, \forall i, j \in [1, \ldots, G], i \neq j$, which means that there are no common demands between any two clients that belong to different groups.

Assuming that such a subset comprises Num items, the demand probability p_i for each item in place i in that subset is computed using the Zipf distribution, where θ_1 is the access skew coefficient. This distribution has also been used in other papers that deal with data broadcasting [2]–[5], [8]. We have

$$p_i = c\left(\frac{1}{i}\right)^{\theta_1} \quad \text{where } c = 1 \Big/ \sum_k \left(\frac{1}{k}\right)^{\theta_1} \quad k \in [1, \ldots, \text{Num}]. \quad (7)$$

We consider that NumS_g, $1 \leq g \leq G$, is the number of items that constitute S_g. The number of items per subset is given in any of the following two forms: 1) the same for each subset S_g and equal to M/G or 2) random for each subset, with NumS_1 + Num$S_2 + \cdots +$ Num$S_G = M$.

In the case of environments that dynamically change their group sizes, we assume that the size of the groups changes every ChGS broadcasts, whereas in environments with different priority levels, $A(1), A(2), \ldots, A(G)$ is considered to be the priority coefficient of each group. Finally, we define as "emigration probability" the probability that a client is considered to be a member of a different group from the one in which he/she lies because of the nonupdated GPS data in the case of indoor location services. The "emigration probability" of each client is computed as follows.

Considering a client that belongs to Group i, the probability that the client is considered to be a member of Group i is equal to 0.88, the probability that the client is considered to be a member of Group $(i+1)$ is 0.1, the probability that the client is considered to be a member of Group $(i+2)$ is 0.01, whereas the probability that the client is considered to be a member of Group $(i+3)$ is also 0.01.

C. Simulation Environment

We performed our experiments with an event-driven simulator that was coded in Matlab. The server has no *a priori* knowledge of item demands; thus, initially, all items have the same demand probability. The broadcast server broadcasts according to the automaton's item probabilities p'_i and the cost function (2) and updates the p'_i estimations after receiving and setting weights to the feedbacks. The broadcasts are subject to reception errors, with unrecoverable errors per instance of an item occurring according to a Poisson process with rate λ, which was also used in [3] and [5]. Thus, $E = 1 - e^{-\lambda}$ is the probability that an item is received with an unrecoverable error.

The simulation runs until the server broadcasts N items. The overhead due to the duration of the feedbacks and the signal propagation delay is defined using the parameter Ouh.

D. Simulation Results

The simulation results in this section are obtained with the following eight values to the parameters

1) NumCl $= 5000$.
2) $N = 200000$.
3) ChGS $= 2000$.
4) SrvCP $= 3000$.
5) Ouh $= 10^{-3}$.
6) $\lambda = 0.1$.
7) $L = 0.15$.
8) $a = 10^{-4}$.

The figures depict the results of our simulations in many different environments, with their characteristics being summarized in Table I.

Fig. 1(a) and (b) display the mean response time per group and the percentage of the server's broadcasts per group as they have previously been defined, respectively, for network N_1. Similar results are presented in Figs. 1(c), 2(a), 3(a)–(c), 4(a)–(c), 5(a)–(c), and 6(a) for networks N_4, N_6, N_7, N_8, N_9, and N_{10}, respectively. Fig. 2(b) and (c) depict the percentage of the server's broadcasts per group for networks N_2, N_3, N_4,

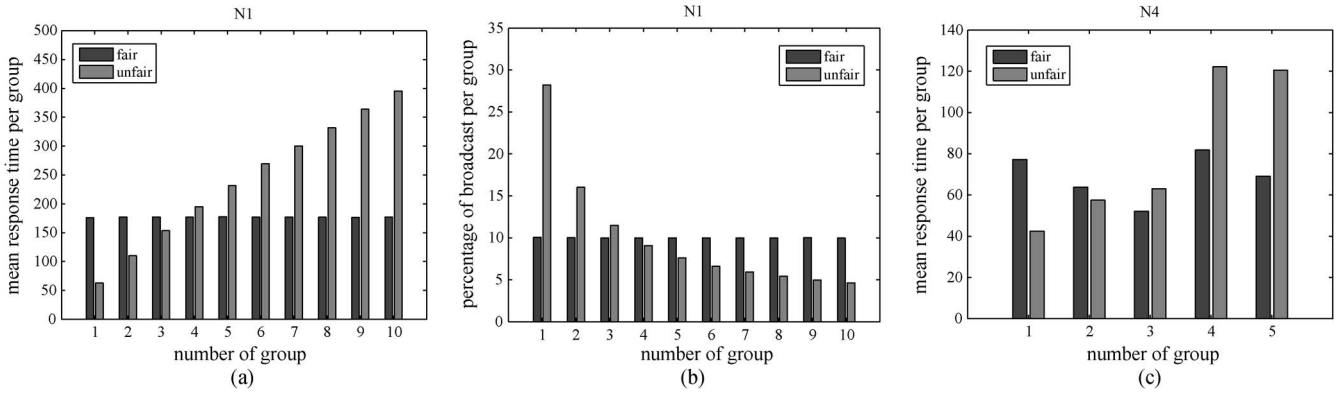

Fig. 1. (a) Mean response time per group in N_1. (b) Percentage of broadcasts per group in network N_1. (c) Mean response time per group in N_4.

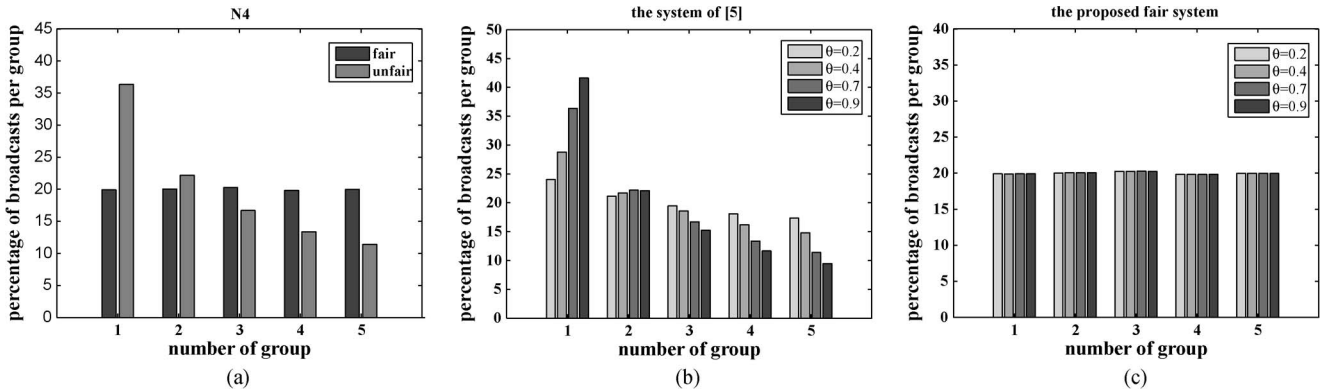

Fig. 2. (a) Percentage of broadcasts per group in network N_4. (b) Percentage of broadcasts per group at the push system that does not provide fairness for four different networks, i.e., N_2, N_3, N_4, and N_5. (c) Percentage of broadcasts per group at the proposed fair push system for four different networks, i.e., N_2, N_3, N_4, and N_5.

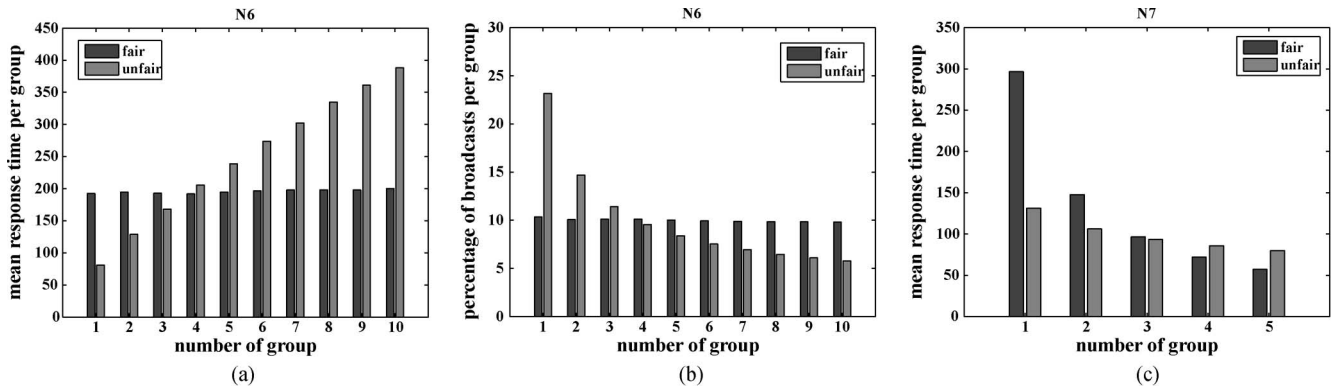

Fig. 3. (a) Mean response time per group in N_6. (b) Percentage of broadcasts per group in network N_6. (c) Mean response time per group in N_7.

and N_5 at the proposed fair push system compared with the system in [5], respectively. Fig. 6(b) depicts the overall mean response time of the system for networks N_1, N_4, N_6, N_7, N_8, N_9, and N_{10}. Observing the figures, we conclude the following results.

The simulation results in Fig. 1(a) and (b) confirm that the proposed system provides fairness. Using an equal number of items per subset $S_g(\text{Num}S_1 = \text{Num}S_2 = \cdots = \text{Num}S_{10} = 20)$ to avoid any dependences of our results on the number of items per subset S_g, we can observe that, in the proposed fair push system, the mean response time per group and the percentage of broadcasts per group are equal and independent of the number of clients per group, whereas in the system in [5], these values are not equal.

Figs. 1(c) and 2(a) show the fairness that this approach achieves in a more realistic environment, where the number of items per subset S_g is not equal $(\text{Num}S_1 = 23, \text{Num}S_2 = 18, \text{Num}S_3 = 14, \text{Num}S_4 = 25, \text{ and } \text{Num}S_5 = 20)$. In Fig. 1(c), the values of the mean response time per group, which are very low at the system in [5], increase, whereas the ones that are high decrease. Thus, all mean-response-time values approach each other, and the system becomes fairer. It is also noticeable that, at the fair push system, the percentage of broadcasts per group is almost equal for each group [see Fig. 2(a)], which

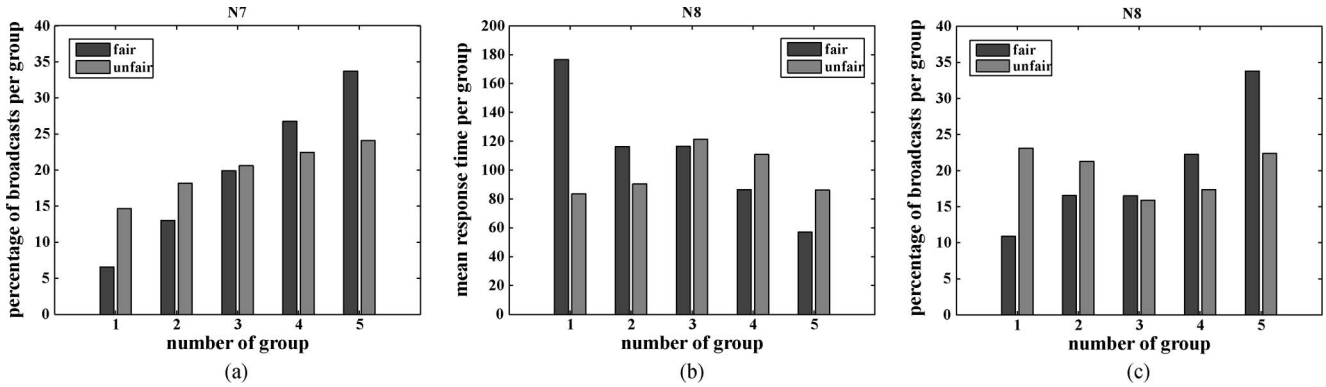

Fig. 4. (a) Percentage of broadcasts per group in network N_7. (b) Mean response time per group in N_8. (c) Percentage of broadcasts per group in network N_8.

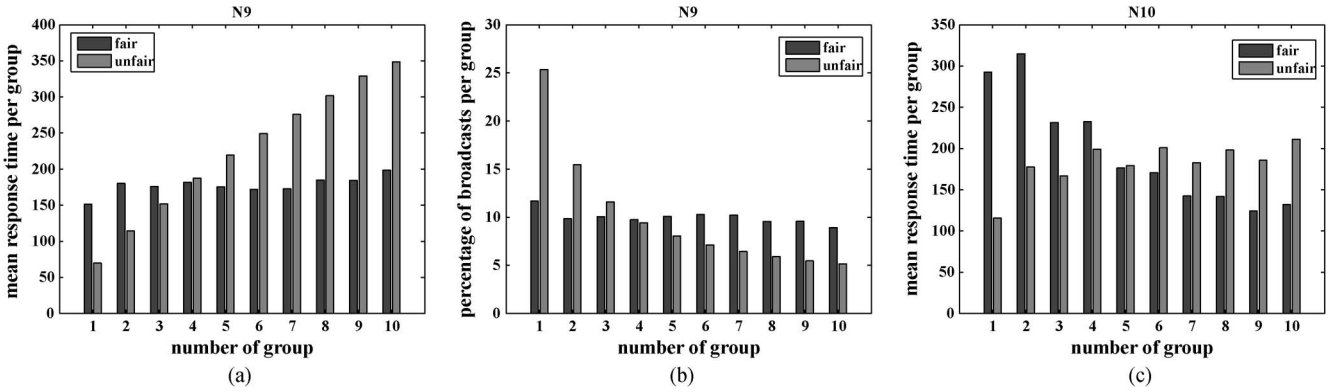

Fig. 5. (a) Mean response time per group in N_9. (b) Percentage of broadcasts per group in network N_9. (c) Mean response time per group in N_{10}.

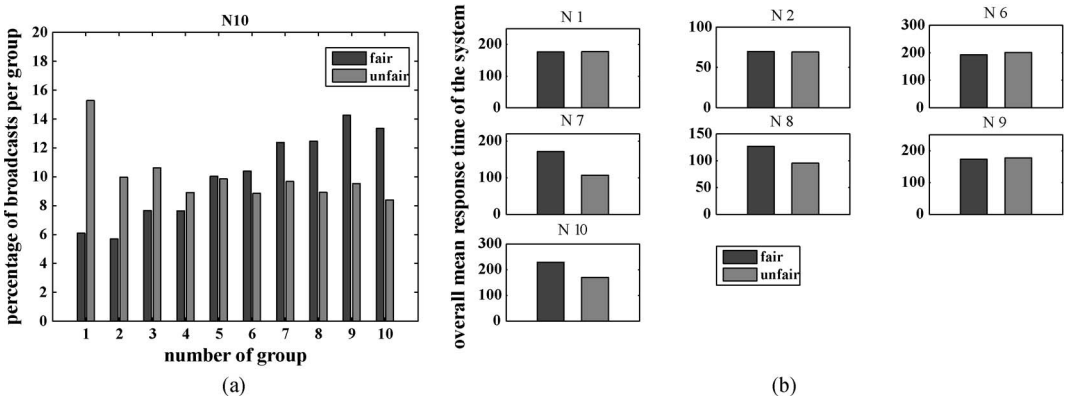

Fig. 6. (a) Percentage of broadcasts per group in network N_{10}. (b) Overall mean response time of the systems in networks N_1, N_4, N_6, N_7, N_8, N_9, and N_{10}.

does not hold for the system in [5], in which the percentage of broadcasts is not equal for each group and depends on the size of each group. In particular, the large groups (e.g., Group 1) have much higher percentage of broadcasts at the expense of the small groups (e.g., Group 5), which have very low percentage of broadcasts at the system in [5].

At the fair push system, the number of server's broadcasts is almost equally split among the groups of clients using either an equal number of items per subset [see Fig. 1(b)] or a random number of items per subset [see Fig. 2(a)].

As far as the values of the mean response time at the fair push system in network N_4 are concerned [see Fig. 1(c)], we observe that these values are not equal to each other as they were in N_1 [see Fig. 1(a)]. This result is predictable, because the demands of the clients of a group that refer to a subset of great range (e.g., Group 4, Num$S_4 = 25$) vary much more than the ones of a group that refers to a subset of small range (e.g., Group 3, Num$S_3 = 14$). In such an environment, the server finds difficulty in satisfying the majority of the demands, and thus, the learning automaton, based on the feedbacks that it receives, is adapted a little slower. As a result, the clients of a group that demand items from a subset of small range (e.g., Group 3, Num$S_3 = 14$) have smaller response times than the clients of a group that demand items from a subset of great range (e.g., Group 4, Num$S_4 = 25$). This situation does not appear in N_1, where all the values of the mean response time per group are equal, because all the subsets of items are equal to each other.

In the system in [5], the percentage of the broadcasts is not equally split among the groups. In particular, the percentage is high at the large groups at the expense of the percentage of the small ones. It is also important to mention that, as the values of θ increase, the system becomes even more unfair for the groups with few members, because their percentage of broadcasts dramatically decreases [see Fig. 2(b); $\theta = 0.9$]. In contrast, at the proposed fair push system, for each value of θ, the percentage of broadcasts per group is almost the same [see Fig. 2(c)].

Fig. 3(a) and (b) depict the fairness that this approach achieves in an environment where the group sizes dynamically change.

Figs. 3(c) and 4(a)–(c) present the simulation results in environments with different priority levels among groups. In these figures, the proposed priority mechanism, which is characterized as "fair," is compared with the simple multiplying of the feedbacks by each priority coefficient, which is characterized as "unfair." In network N_7 [see Figs. 3(c) and 4(a)], there are five different priority levels (i.e., $A_1 = 1, A_2 = 2, A_3 = 3, A_4 = 4,$ and $A_5 = 5$), with Group 1 having the lowest $(A_1 = 1)$. Due to its low priority, Group 1 has the highest mean response time per group and the lowest percentage of broadcasts per group among all the groups. Using the proposed priority mechanism, Group 2, which has double the priority of Group 1 $(A_2 = 2)$, is served with double the frequency of Group 1. Thus, Group 2's mean response time is half that of Group 1, and its percentage of broadcasts is double that of Group 1. Respectively, Group 3 $(A_3 = 3)$ has one third the mean response time and triple the percentage of broadcasts of those of Group 1. Observing Figs. 3(c) and 4(a), it is obvious that the "unfair" priority mechanism cannot efficiently be adapted to the different priorities levels of the system.

Fig. 4(b) and (c) depict the simulation results of network N_8, which also uses priorities. It is noticeable that, despite the unequal sizes of Groups 2 and 3, the proposed priority mechanism achieves the same performance (equal mean response time per group and equal percentage of broadcasts per group) for these two groups of the same priority level $(A_2 = A_3 = 1.5)$, which is not achieved by the "unfair" priority mechanism.

Fig. 5(a) and (b) confirm that the proposed mechanism that provides fairness efficiently works in environments with "emigration probability" due to the nonupdated GPS data in the case of indoor location services.

Figs. 5(c) and 6(a) present the simulation results in network N_{10}, which combines both priorities and "emigration probability." In this network, each successive pair of groups have the same priority, with the first two and last two groups having the lowest and the highest priority, respectively. However, the absolute multiplying of the values of the percentage of broadcasts or the mean response time per group based on the priorities is not achieved in this network as it was achieved in networks N_7 and N_8. This result is expected due to the "emigration probability" that affects the simulation results. Nevertheless, it is noticeable that, using the proposed "fair" priority mechanism, the groups with the same priority have the same or almost the same percentage of broadcasts or mean response time per group (e.g., Group 1 with 2, Group 3 with 4, and Group 5 with 6). On the contrary, observing these figures, it is obvious that the "unfair" priority mechanism cannot efficiently be adapted to the different priority levels of the system.

Fig. 6(b) confirms that this new approach does not significantly affect the overall mean response time (overall performance) of the system. Apart from networks with different priority levels (e.g., N_7, N_8, and N_{10}) where there is a relative increase in the overall response time of the system, no significant effect is observed in the other networks.

V. Conclusion

This paper has proposed a push system for environments with unequal group sizes and different priority-level client demands. Using different weight coefficients for the clients' feedbacks of each group, the server succeeds in providing fairness to the response time and the percentage of broadcasts among groups with different population, considering the size and the priority level of each group.

Acknowledgment

The authors would like to thank Prof. T. Yioultsis, who willingly contributed to the improvement of this paper with his valuable comments and suggestions.

References

[1] D. Aksou and M. Frankin, "RxW: A scheduling approach for large-scale on-demand data broadcast," *ACM/IEEE Trans. Netw.*, vol. 7, no. 6, pp. 846–860, Dec. 1999.

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

[3] N. H. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments," *Wirel. Netw.*, vol. 2, no. 3, pp. 171–182, May 1999.

[4] C. J. Su and L. Tassiulas, "Broadcast scheduling for information distribution," in *Proc. IEEE INFOCOM*, May 1997, pp. 109–117.

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

[6] P. Nicopolitidis, G. I. Papadimitriou, and A. S. Pomportsis, "Multiple-antenna data broadcasting for environments with locality of demand," *IEEE Trans. Veh. Technol.*, vol. 56, pt. 1, no. 5, pp. 2807–2816, Sep. 2007.

[7] K. Stathatos, N. Roussopoulos, and J. S. Baras, "Adaptive broadcasts in hybrid networks," in *Proc. VLDB*, 1997, pp. 326–335.

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

[9] X. Wu and V. C. S. Lee, "Preemptive maximum stretch optimization scheduling for wireless on-demand data broadcast," in *Proc. IEEE Int. Database Eng. Appl. Symp.*, 2004, pp. 413–418.

[10] C.-L. Hu and M.-S. Chen, "Adaptive balanced hybrid data delivery for multichannel data broadcast," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, pp. 960–964.

[11] P. Triantafillou, R. Harpantidou, and M. Paterakis, "High-performance data broadcasting systems," *Mobile Netw. Appl.*, vol. 7, no. 4, pp. 279–290, Aug. 2002.

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

[13] G. I. Papadimitriou, "A new approach to the design of reinforcement schemes for learning automata: Stochastic estimator learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 6, no. 4, pp. 649–654, Aug. 1994.

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

[15] K. Schreiner, "Where we at? Mobile phones bring GPS to the masses," *IEEE Comput. Graph. Appl.*, vol. 27, no. 3, pp. 6–11, May–Jun. 2007.
[16] N. Ueda, Y. Nakanishi, S. Matsukawa, and M. Motoe, "Developing a GIS using a mobile phone equipped with a camera and a GPS, and its exhibitions," in *Proc. IEEE 24th ICDCSW*, pp. 414–417.
[17] Y. Nakajima, H. Shiina, S. Yamane, and T. Ishida, "Disaster evacuation guide: Using a massively multiagent server and GPS mobile phones," in *Proc. IEEE SAINT*, p. 2.
[18] R. Matos, D. F. Santos, J. E. Sanguino, and A. Rodrigues, "A GPS-based mobile coordinated positioning system for firefighting scenarios," in *Proc. 1st Int. Conf. MCWC*, 2006, pp. 209–214.
[19] C. E. Palazzi, "Buddy-finder: A proposal for a novel entertainment application for GSM," in *Proc. IEEE Commun. Soc. Globecom Workshops*, pp. 540–543.
[20] K. S. Gilhousen, I. M. Jacobs, R. Padovani, A. J. Viterbi, L. A. Weaver, and C. E. Wheatley, III, "On the capacity of a cellular CDMA system," *IEEE Trans. Veh. Technol.*, vol. 40, no. 2, pp. 303–312, May 1991.
[21] C. Y. Lee, "Overview of cellular CDMA," *IEEE Trans. Veh. Technol.*, vol. 40, no. 2, pp. 291–302, May 1991.
[22] M. H. Ammar and J. W. Wong, "On the optimality cyclic transmission in teletext systems," *IEEE Trans. Commun.*, vol. COM-35, no. 1, pp. 68–73, Jan. 1987.
[23] R. Jain and J. Werth, "Airdisks and airRAID: Modeling and scheduling periodic wireless data broadcast (extended abstract)," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 4, pp. 23–28, Sep. 1995.
[24] E. Snekkenes, "Concepts for personal location privacy policies," in *Proc. 3rd ACM Conf. Electron. Commerce*, 2001, pp. 48–57.

Vasiliki L. Kakali received the B.S. degree in computer science from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2004. She is currently working toward the Ph.D. degree with the Department of Informatics, Aristotle University of Thessaloniki.

Her research interests include wireless networks, data broadcasting, and mobile communications.

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

In 1997, he joined the faculty of the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece, where he is currently an Associate Professor. His research interests include optical networks and wireless networks. He is a coauthor of three books published by Wiley. He is the author or a coauthor of 80 journal papers and 90 conference proceedings.

Dr. Papadimitriou is an Associate Editor for five IEEE Journals.

Petros Nicopolitidis (M'09) received the B.S. and Ph.D. degrees in computer science from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1998 and 2002, respectively.

Since 2004, he has been a Lecturer with the Department of Informatics, Aristotle University of Thessaloniki. His research interests include wireless networks and mobile communications. He is a coauthor of *Wireless Networks* (Wiley, 2003).

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

He is currently a Professor with the Department of Informatics, Aristotle University, Thessaloniki. He is a coauthor of *Optical Switching* (Wiley, 2007), *Multiwavelength Optical LANs* (Wiley, 2003), and *Optical Networks* (Wiley, 2003). His research interests include computer networks, computer architecture, parallel and distributed computer systems, and multimedia systems.