A Novel Adaptive Framework for Wireless Push Systems based on Distributed Learning Automata

V. L. Kakali, P. G. Sarigiannidis, *Member, IEEE,* G. I. Papadimitriou, *Senior Member, IEEE,* and A. S. Pomportsis

Abstract—**A novel adaptive scheme for wireless push systems is presented in this paper. In this wireless environment two entities play the most important role: the server side and the client side that is connected to the system. The server side is responsible to broadcast an item per transmission in order to satisfy the clients' requests. The performance of the server side depends on item selections. Hence, the server broadcasts an item and the clients are satisfied if the transmitted item was the desired one. In this work, a set of learning automata try to estimate the client demands in a distributed manner. More specifically, an autonomous learning automaton is utilized on each client group, since the clients are gathered into groups based on their location. The output of each automaton is combined in order to produce a well-performed transmission schedule. Concurrently, a round robin phase is adopted, giving the opportunity to the non-popular items to be transmitted. In this manner, the various client demands are treated fairly. The introduced technique is compared with a centralized adaptive scheme and the results indicate that the proposed scheduling framework outperforms the centralized one, in terms of response time and fairness.**

Index Terms—**Distributed Learning Automata, Fairness, Locality of Demand, Wireless Push Systems.**

I. Introduction

Data broadcasting has emerged as an efficient way of delivering information over wireless networks (i.e. traffic information, weather information, news distribution). In such applications, client preferences are usually overlapping and the broadcast of a single information item will likely satisfy a large number of clients. The three major approaches for designing broadcast schedules is the pull, the push and the hybrid one. In pull systems, (e.g. [1, 2, 3, 4]), data items are broadcast in response to explicit requests received from clients. Pull can be used either for unicast or for broadcast. When used for broadcast, pull is also referred to as interactive or on-demand broadcast [5].

The advantage of these systems is their adaptive to dynamic client demands. Though, they are not scalable for large client populations as client requests will either collide with each other or saturate the server. In push systems (e.g. [6, 7, 8]), the server is considered to have an a-priori estimation of the client demands and schedules its broadcasts according to these estimates of the demand per information item. On the contrary to the pull approach, the "pure" push systems provide high scalability and client hardware simplicity but are unable to operate efficiently in environments with a-priori unknown and dynamic client demands.

The adaptive push system of [9, 10, 11] though, achieves efficient operation in such environments using a learning automaton [12] at the broadcast server in order to provide adaptivity. The server conserves a database of the items that are broadcast while the learning automaton contains the server's estimate of the demand probability for each item. After each item broadcast, every satisfied client sends to the server a feedback. Using the clients' feedbacks, the server's automaton continuously adapts to the overall client population demands in order to reflect the overall popularity of each data item. Finally, hybrid systems (e.g. [13, 14, 15]) 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 of [9].

The adaptive push system of [9] uses a single centralized automaton to adapt its scheduled broadcasts to the clients' demands. Though, most applications are characterized by locality of demand which means that clients are gathered into groups, each one located at a different region. Members of each group have similar demands for information items, different from the demands of clients at other groups (e.g. traffic information) while the size of each group (number of clients per group) varies as there are groups of large population and groups of small one. Thus, a centralized automaton is not able to target on the explicit needs and features of each group as it considers all the clients as a single group.

The proposed scheme alters the operation of designing the broadcast schedules by applying a distributed learning automata scheme [16] instead of a centralised one. The proposed DIstributed Automata Scheme (DIAS) framework targets on accurate estimations regarding the clients' demands via a distributed operation of the applied automata. In this manner, the scheme takes into account the locality of item demands per group. Furthermore, the novel scheme focuses on each group separately taking into consideration the size of each group, regarding the choice of the forthcoming broadcasts. This means that the server side tries to determine the client demands per group by applying a learning automaton on each group of clients distinctly, contrary to a unique centralized learning automaton for the whole system. This design leads to a more sensitive framework than the

centralized one, since the client demands are treated with an efficient and fair way. Simulation results indicate that the distributed nature of the proposed scheme results in better performance in terms of response time and fairness than the centralised one.

The distributed automata have been widely used in the literature. Indicatively, it is mentioned their use on algorithms for solving multi-agent markov decision processes [17], for learning minimum delay paths in service overlay networks [18], in distributed fuzzy logic processor training [19] and for function optimization problems [20].

The remainder of the paper is organised as follows: Section II presents previous schemes for the push approach, section III analyzes the system architecture and presents the proposed scheme. Simulation results are presented in Section IV. Finally, Section V concludes the paper.

II. Related Work

The method for data broadcasting in push systems presented in [7] is based on two statements that optimize the system performance: (a) schedules with minimum overall mean access time are produced when the intervals between successive instances of the same item are equal, (b) 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 that an item of length l_i is received with an unrecoverable error.

The algorithm of [7] operates as follows: The server contains a database of *K* information items, and *p* is the vector of the estimated probability demand per item. Assuming that *t* is the current time and *T(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):

$$CF(i) = (t - T(i))^2 \frac{p_i}{l_i}(1+E(l_i))/(1-E(l_i)),\ 1 \le i \le K \quad (1)$$

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, *T(i)* is initialized to *-1* and if the maximum value of *CF(i)* is given by more than one items, the algorithm selects one of them arbitrarily. Upon the broadcast of item *i* at time *t*, *T(i)* is changed so that *T(i)=t*.

The main disadvantage of this method is its lack of adaptivity and thus its inefficiency in environments with a-priori unknown client demands. The adaptive push system of [9] achieves efficient operation in such environments using a single Learning Automaton at the broadcast server in order to provide adaptivity. After each item broadcast, every satisfied client sends to the server a feedback ("vote"). Using the client feedbacks-votes, the server's automaton continuously adapts to the overall client population demands in order to reflect the overall popularity of each data item. The operation of the centralised learning automaton is given by the equation below:

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

It holds that $L, a \in (0,1)$ and $p_i'(k) \in (a,1)$, $\forall\ i \in [1,...,K]$, where *K* is the number of the server's items, p' is the server's estimated demand probability vector, *L* is a parameter that governs the speed of the automaton convergence and the role of parameter a is to prevent the probabilities of unpopular items from taking values in the neighbourhood of zero in order 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]. $b(k) = 1 - \frac{number\ of\ received\ feedbacks}{total\ number\ of\ clients}$ is the system environmental response that is triggered after the server's k^{th} transmission.

In the remainder of the paper this method will be referred as the "centralized system" and will be used for the comparison and validation of the DIAS.

III. The DIstributed Automata Scheme (DIAS)

A. The System Architecture

The system consists of a base station and mobile clients. The client topology is characterized by locality of demand. *P* denotes the client population and *G* the number of groups that are gathered into. The members of each group may demand *M* information items. Clients are considered equipped with GPS receivers. As nowadays many PDAs are typically equipped with GPS receivers, there is no need for complex client equipment [21, 22, 23, 24, 25].

The base station consists of one omni-directional antenna and a database of G×M different items. The server side is also equipped with *G* distributed automata (one for each group) in order to adapt the broadcast scheduling to the explicit group demands. The base station broadcasts the server's data items while clients respond ("vote") to the server's broadcasts.

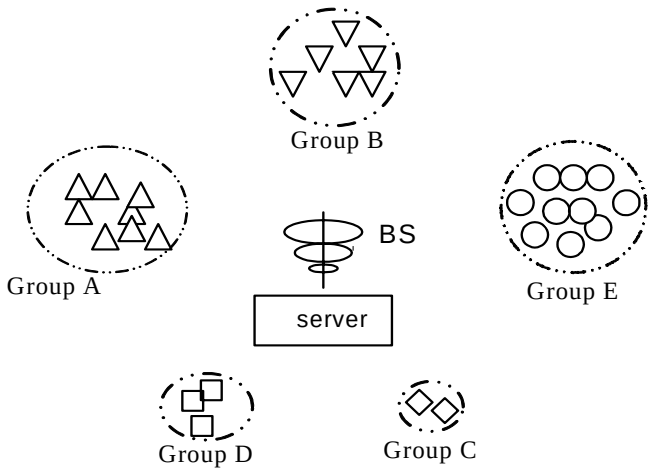

Fig. 1. The system architecture.

For the uplink communication, a CDMA coding has been chosen [26, 27]. After the broadcast of an item that has been expected from a client, the client's software application sends automatically its "vote" (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 (rake receiver) 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.

Broadcasts are organized into transmission frames. Each frame comprises of three phases, the learning phase, the calculation phase and the schedule phase. The system architecture is depicted in Fig. 1.

B. The Learning phase

Each Learning Automaton contains the server's estimate $p'_{i,g}$ of the demand probability $p_{i,g}$ for each item *i* among the set of the items that the server broadcasts for the specific group *g*. Thus, for each group *g*, it holds that $\sum_{i=1}^{M} p'_{i,g} = 1$, where *M* is the number of items that refers to each group.

During the learning phase, the broadcast follows a round robin manner sending each item once. Hence, in this phase, the scheme gives to the clients the opportunity to be satisfied at least once per frame. This action improves the fairness factor of the system operation. The server executes *GxM* broadcasts sending each item once without taking into account the server's estimate $p'_{i,g}$ of the demand probability and thus balancing the percentage of broadcasts at each group. Figure 2 depicts the learning phase in system with two groups.

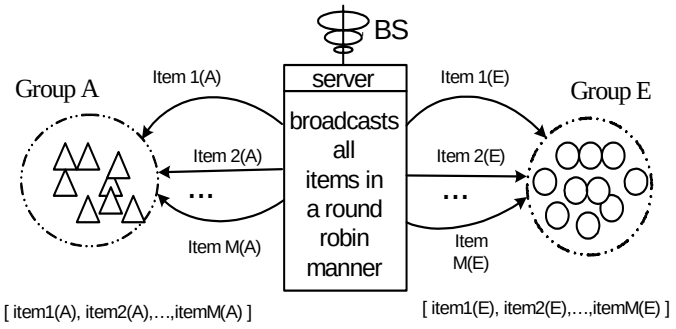

Fig. 2. The learning phase

After the transmission of item *i* of group *g*, each satisfied client transmits a feedback (e.g. one bit), using Code Division Multiple Access (CDMA) as described in subsection III.A.

After the reception of the feedbacks, the base station decodes them and the server updates the distributed demand probability vector p'_g according to the probability updating scheme that is described below.

When there are satisfied clients, the probability of the transmitted item *i* increases while the probabilities of all the rest items decrease. Considering that the server's k^{th} transmission is item *i* of group *g*, the probability vector p'_g is updated according to equation (3):

$$\begin{aligned} p'_{j,g}(k+1) &= p'_{j,g}(k) - L(1-b_g(k))\,(p'_{j,g}(k)-a),\ \forall j \neq i \\ p'_{i,g}(k+1) &= p'_{i,g}(k) + L(1-b_g(k)) \sum_{i\neq j} (p'_{j,g}(k)-a) \end{aligned} \quad (3)$$

where $p'_{i,g}(k) \in (a,1),\ \forall\ i \in [1,...,M]$, $L, a \in (0,1)$, $b_g(k) = 1 - \frac{number\ of\ satisfied\ clients\ of\ g}{number\ of\ clients\ of\ g}$, L is a parameter that governs the speed of the automaton convergence while parameter *a* prevents the probabilities of unpopular items from taking values around zero, as some items even if unpopular may be demanded by some clients. The $b_g(k)$ parameter is the system's environmental response of group *g* after the reception of the client feedbacks of group *g*. A value of $b_g(k)$ that equals one represents the case where no client feedback is received.

The server is aware of the population of each group *g* using the following mechanism: The server, which has a priori knowledge of the geographic locations (co-ordinates) where its services are offered, sends one control packet for each one of these locations, asking of the clients that belong to this location to send back a feedback. The client's GPS receiver detects its co-ordinates and the client sends its feedback 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 received from each area.

A vector *F* of size equal to the number of groups *G* is considered. After each broadcast, the number of received feedbacks from the group *g* is added in *F(g)*. At the end of the learning phase, the value of the mean number of feedbacks per group is stored in the F' vector in order to be used for the calculation phase as follows $F'_g = \frac{F_g}{M}$.

C. The calculation phase

The number of broadcasts for each group of the next phase is formed based on the formula below:

$S_g = \frac{F'_g}{\sum_{i=1}^{G} F'_g} \cdot G \cdot M$, where S_g is the number of the next broadcasts dedicated to each group.

Based on the formula above the *GxM* broadcasts of the next phase are distributed to the groups according to the proportion

of the mean number of feedbacks per group to the total number of feedbacks ($\sum_{i=1}^{G} F_g^{'}$).

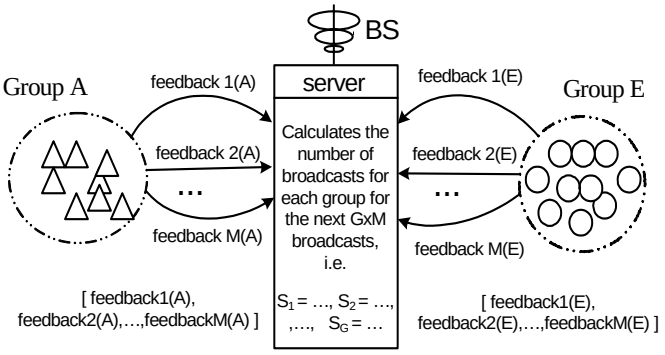

Fig. 3. The calculation phase.

D. The schedule phase

In this phase, the broadcasts are scheduled based on the previously calculated *S* vector. According to *S* vector, a number of broadcasts equal to *sum(S)* are scheduled. The scheduling algorithm is defined as follows:

Scheduling Algorithm

//begin of scheduling algorithm

while (sum(S)>0)

maximum = max(S);

- *selected group = the index of maximum in vector S*

- *broadcast to the "selected group" the item i that maximizes the cost function CF presented below*

- *update* $p'_{selected\ group}$

- *S(selected group) = S(selected group) – 1*

end while

// end of the scheduling algorithm

The server selects to broadcast to the "selected group" the item i that maximizes the cost function *CF(i)* [7, 9]

$$CF(i) = (T - R(i))^2 \frac{p_{i,g}^{'}}{l_i} \frac{(1+E(l_i))}{(1-E(l_i))},\ 1 \le i \le M \qquad (4)$$

where T is the current time, *R(i)* is the time when item i was last broadcast, $p_{i,g}^{'}$ is the server's estimated demand probability for item i, l_i is the item's length, $E(l_i)$ is the probability than an item of length l_i is received with an unrecoverable error and M is the number of the data items of group g. For items that have not been previously broadcast *R(i)* is initialized to *-1*.

Figure 4 displays the schedule phase as described above in system with two groups.

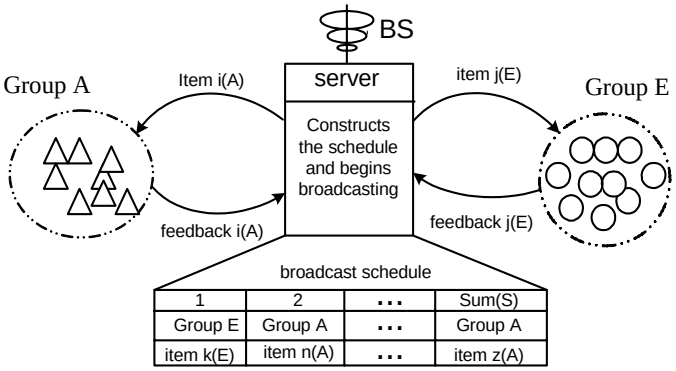

Fig. 4. The schedule phase.

IV. The Simulation Environment

A. Simulation Environment

The server contains a database of K=G x M. the item length l is considered to be equal to the unit for each item for simplicity reasons, as the item length is not an affecting issue for the distributed simulation environment. A population of P clients is considered. Clients are grouped into G groups with each group being located at a different region and having different item demands. In order to model groups of clients with different group sizes, the size of each group is computed via the Zipf distribution. Thus, the number of clients in group x, *Size(x)*, $1 \le x \le G$, is:

$$Size(x) = c\left(\frac{1}{x}\right)^{\theta} \cdot P \qquad (5)$$

where $c = \frac{1}{\sum_k \left(\frac{1}{k}\right)^{\theta}}$, $k \in [1..G]$ and θ is a parameter named skew coefficient. For $\theta = 0$, the Zipf distribution produces equally-sized groups while for large values of θ, the Zipf distribution produces increasingly skewed patterns.

Any client belonging to group x is interested in the same subset K_x of server's data items. All items outside this subset have a zero demand probability at the clients of the group. Moreover, $K_{x1} \neq K_{x2}$, $\forall x1, x2 \in [1..G]$, $x1 \neq x2$, which means that there are not any common demands between any two clients belonging to different groups.

In each subset, the client demand probability p_x for each item in place x in that subset, is computed also via the Zipf

distribution. This distribution has also been used in other papers that deal with data broadcasting [6, 7, 9, 11]

$$p_x = c\left(\frac{1}{x}\right)^{\theta_1} \qquad (6)$$

where $c = \frac{1}{\sum_k \left(\frac{1}{k}\right)^{\theta_1}}$, $k \in [1..\mathrm{NumK}_x]$, $NumK_x$ is the number of items that comprises the subset K_x, and θ_1 is the skew coefficient. After each broadcast, the satisfied clients (the ones that were waiting for the broadcast item) refresh immediately their demands while the "unsatisfied" clients persist to their demands until they get satisfied. Finally, a number of *G* distributed automata are utilized at the server, as they are described in section III.

The broadcasts are subject to reception errors, with unrecoverable errors per instance of an item occurring, according to a Poisson process with rate λ. Thus, $E = 1 - e^{-\lambda}$ is the probability that an item is received with an unrecoverable error.

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

The DIAS is compared with the centralized system of [9] where the server uses one centralised learning automaton to schedule its forthcoming broadcasts.

The performance of the compared schemes is measured in terms of response time and percentage of broadcasts per group. The response time is the mean time that passes until the client receives one item that expects and is defined as follows:

$$ResTime_j = \frac{total\ time\ of\ the\ Broad\ broadcasts}{number\ of\ satisfied\ broadcasts\ of\ client_j}, \quad \text{where}$$

$1 \le j \le P$. In this manner, the response time of the entire system is given by $\frac{\sum_{j=1}^{P} ResTime_j}{P}$. Moreover, 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 and is given by:

$$PercBroad_j = \frac{num\ of\ broadcast\ items\ of\ group_j}{Broad} \cdot 100, \quad \text{where}$$

$1 \le j \le G$.

B. Simulation Results

The experiments are performed in a simulator coded in Matlab. The simulation results presented in this section are obtained with the following values to the parameters: *P=1000*, *Broad=100000*, $Ouh = 10^{-3}$, $\lambda = 0.1$, L=0.15 and a=10^{-4}. Table 1 summarizes the characteristics of the networks that have been simulated while figures 5-14 depict the simulation results of these networks.

Table 1. The characteristics of the different simulation environments.

Network	G	M	θ	θ_1
N_1	[5...20]	5	0.9	0.5
N_2	[5…20]	180/G	0.9	0.5
N_3	10	[5…20]	0.9	0.5
N_4	20	5	[0.0...1.0]	0.5
N_5	20	10	0.9	[0.0…1.5]

Figure 5 depicts the response time of network N_1 as the number of groups varies. In these figures the number of items per group (*M*) is stable and equal to 5, while the number of the groups that the clients are gathered into takes values in the interval of 5 to 20. For instance, when the number of groups is 20, the total items are 5×20 = 100. Obviously, the proposed DIAS scheme achieves better performance than the centralised one for different number of groups.

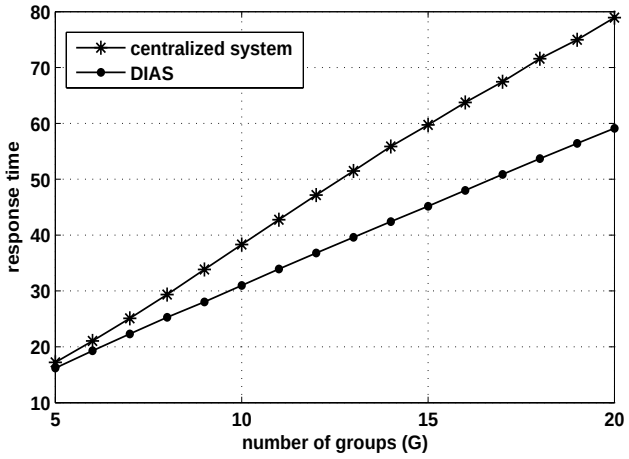

Fig.5. The response time for network N_1.

In Figure 6, the percentage of broadcasts per group of the same network is presented for three different values of system groups. In all the three cases, the distributed scheme seems to be more fair than the centralized one in terms of percentage of broadcasts per group. In the centralized system, the large populated groups (e.g group 1 at fig.6.a, 6.b, 6.c) have much higher percentage of broadcasts at the expense of the small ones which have very low percentage of broadcasts. For example, the percentage of the large group 1of fig 6.a. reaches the 35% of the total server's broadcasts while the percentage of group 10 which is the smaller one is only 2.5%. In the distributed system, the server's broadcasts are scheduled in a more fair matter. This mainly happens because the learning phase is utilized according to a round robin manner. Especially, the values of the percentage per group which are very low at the centralized system increase, whereas the ones that are high decrease. Thus, all the values of the percentage approach to each other and the system becomes more fair.

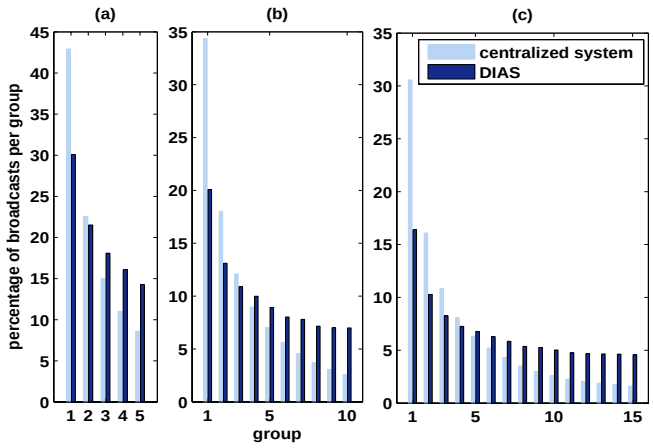

Fig.6. Percentage of broadcasts per group for number of groups equal to (a) 5, (b) 10, (c) 15 for network N_1.

Figure 7 shows the response time of network N_2 versus different values of G (different number of groups) in an environment where the total number database items is stable and equal to 180 independent of the number of groups. Here, the number of items per group M is equal to 180/G. Thus, for number of groups equal to 5, 10 and 15, the number of items per group is 36, 18 and 12 respectively. The response time of the proposed system is significantly lower then that of the centralized one. It is also noticeable the fact that the response time of the distributed system is stable and independent of the number of groups whereas the response time of the centralized one increases with the raise of the number of groups. Apparently, for large values of groups, the centralized system finds difficulty in adapting to the varying demands of many groups. Contrary, the distributed system achieves more accurate estimations of the client demands independent of the number of groups as each automaton targets on a specific group.

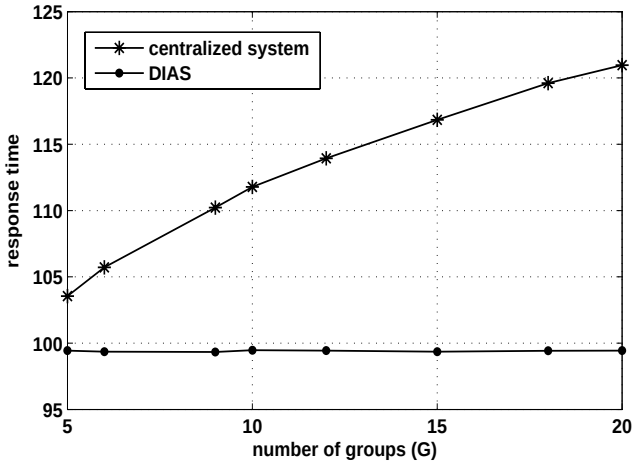

Fig.7. The response time for network N_2 .

Figure 8 shows the percentage of broadcasts per group of the same network N_2 for three values of G ($G=5$, $G=10$, $G=20$). In all the three cases, it stands that the distributed scheme is more fair than the centralized one as the values of the percentage of the groups try to approach to each other.

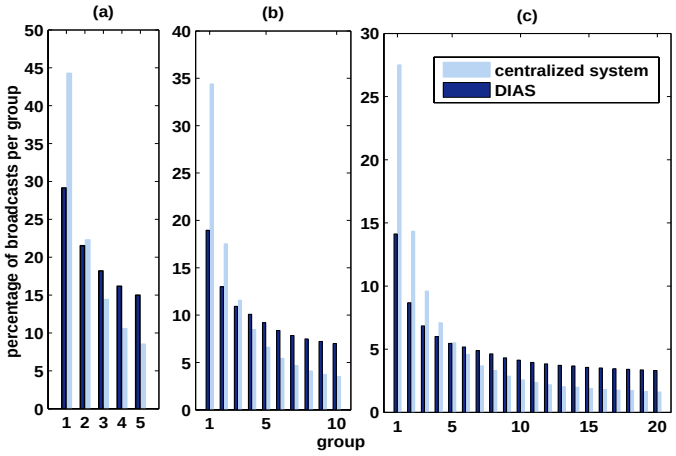

Fig.8. Percentage of broadcasts per group for number of groups equal to (a) 5, (b) 10, (c) 20 for network N_2.

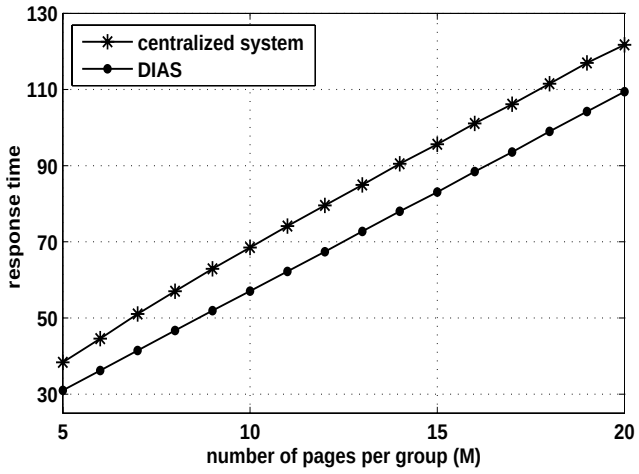

Fig.9. The response time for network N_3.

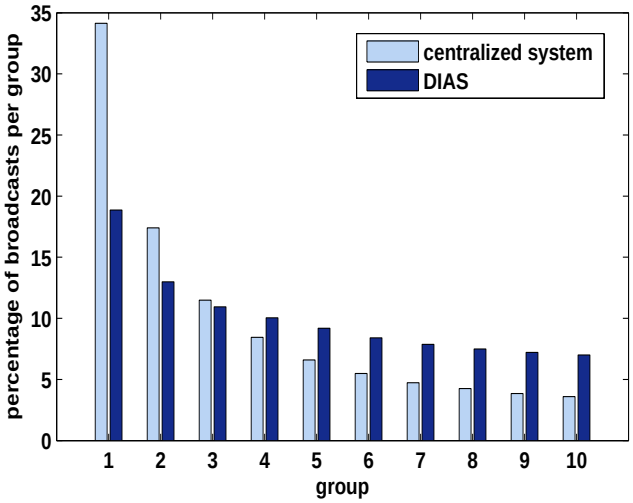

Fig.10. Percentage of broadcasts per group for network N_3.

Figures 9 and 10 confirm the superiority of the proposed scheme at the response time as well as the fairness in terms of

the percentage of broadcasts for different values of *M* (number of items per group). It is worth mentioning that the performance in terms of percentage of broadcasts per group is the same for each value of *M* at both schemes. Thus, it has been chosen indicatively the case of *M=10* to be presented.

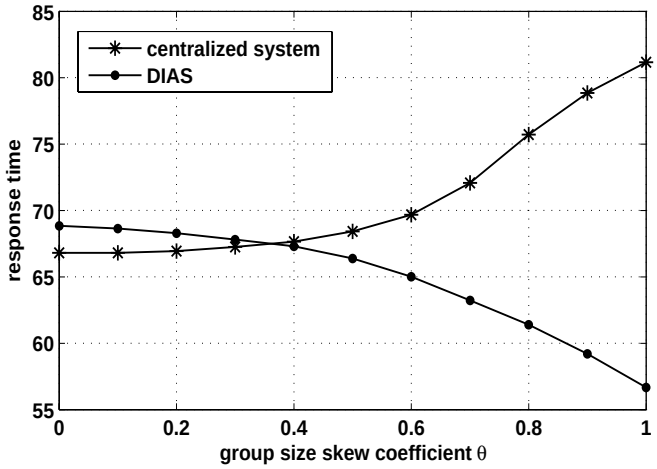

Fig.11. The response time for network N_4.

Figure 11 depicts the response time versus different values of the group size skew coefficient θ of network N_4. Observing the figures, it is obvious that as the θ increases the centralized system follows a growing progress, while the distributed scheme follows a significant decrease. More specifically, for small values of θ (θ<0.4) where the group sizes have very close values, the two systems have similar behavior with the centralized system excelling marginally. Though, for values of θ greater than 0.4, the Zipf distribution produces increasingly skewed patterns. In such a realistic environment with unequal group sizes, the performance of the distributed system outperforms that of the centralized one. Indeed, as the values of θ increases, the performance of the distributed system is enhanced, a fact that does not stand for the centralized system where the performance is degraded.

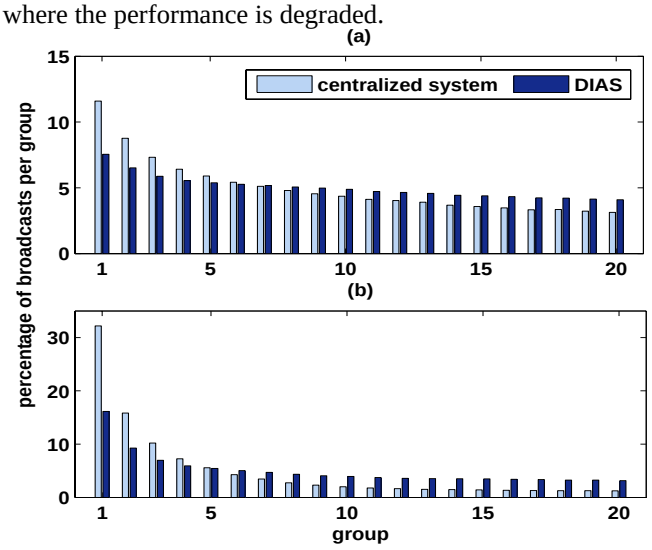

Fig.12. Percentage of broadcasts per group for group size skew coefficient θ to (a) 0.4, (b) 1.0 for network N_4.

In Figure 12, it is presented the percentage of broadcasts per group of N_4 for two values of θ (θ=0.4, θ=1.0). In both cases, it holds that the distributed scheme is more fair than the centralized one. This phenomenon can be explained as the values of the percentage broadcasts of the groups try to approach to each other in compare with the centralized one where the first large group has the majority of the server's broadcasts at the expense of the other ones. It is worth mentioning that the lack of fairness at the centralized scheme is more intense for large values of θ because the group size is intensively skewed. The above is revealed in Figure 12.b. where θ=1.0 and the percentage of the first group with Size(1) = 278 is 32% while the percentage of the last one with Size(20) = 14 is 1.2%. For the distributed system, these values are converged with the high percentage decreasing to 16% while the low one increasing to 3.2%.

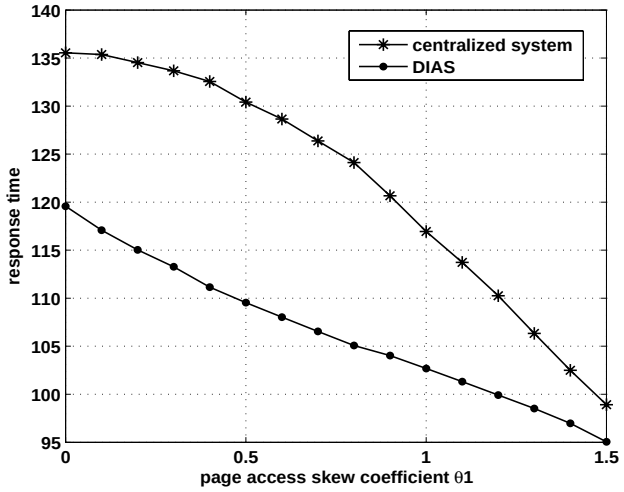

Fig.13. The response time for network N_5.

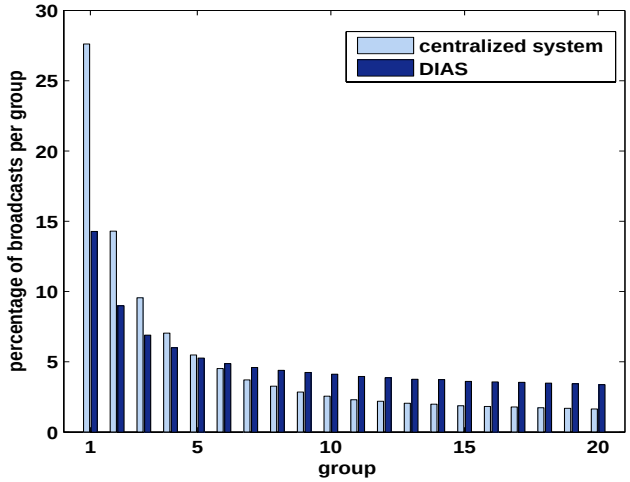

Fig.14. Percentage of broadcasts per group for network N_5.

Figures 13 and 14 present the outperformance of the proposed scheme for different values of θ_1 (page access skew coefficient). It is important to say that the performance in

terms of percentage of broadcasts per group is the same for each value of θ_1 at both schemes. Thus, it has been chosen indicatively the case of θ_1 =0.0 to be presented.

V. Conclusion

A new distributed adaptive scheme for wireless push systems was proposed in this work. The novelty of the introduced scheduling framework has to do with the adoption of a distributed placing of autonomous learning automata. The proposed DIAS scheme seems to be more sensitive concerning the various client demands, by utilizing an autonomous adaptive mechanism for each different region-based client group. Moreover, DIAS introduced a more fair treatment of the less popular client requests, without sacrificing the system performance. This is proved by extensive simulations experiments that point out the superiority of the novel framework compared to the centralized one, in terms of response time and fairness.

VI. References

[1] D.Aksou and M.Frankin, "RxW: A Schedualing Approach for Large-Scale On-Demand Data Broadcast", ACM/IEEE Transactions on Networking vol.7, no.6, pp.846-860, December 1999.

[2] W. Sun, W. Shi and B.Shi, "A Cost Efficient Scheduling Algorithm of On-Demand Broadcasts", Wireless Networks, vol.9, no.3, pp.239-247, May 2003.

[3] Xiao Wu and Victor C. S. Lee, "Preemptive Maximum Stretch Optimization Scheduling for Wireless On–Demand Data Broadcast", IEEE Proceedings of the International Database Engineering and Applications Symposium, 2004.

[4] Peter Triantafillou, R. Harpantidou and M. Paterakis, "High Performance Data Broadcasting Systems", Kluwer Mobile Networks and Applications 7, 279–290, 2002.

[5] Aksoy, D. Leung, M.S.-F, "Pull vs push: a quantitative comparison for data broadcast", IEEE GLOBECOM '04, vol.3, pp 1464- 1468.

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

[7] N.H.Vaidya and S.Hameed, "Scheduling Data Broadcast In Asymmetric Communication Environments", Wireless Networks, vol.2, no.3, pp.171-182, May 1999.

[8] E.Yajima, T. Hara, M. Tsukamoto, S. Nishio, **"**Scheduling and caching strategies for correlated data in push-based information systems**",** ACM SIGAPP Applied Computing Review, Vol. 9 , Issue 1, pp. 22-28, 2001.

[9] P. Nicopolitidis, G. I. Papadimitriou and A. S. Pomportsis, " Using Learning Automata for Adaptive Push - Based Data Broadcasting in Asymmetric Wireless Environments ", IEEE Transactions on Vehicular Technology, vol.51, no.6, pp.1652-1660, November 2002.

[10] P.Nicopolitidis, G.I.Papadimitriou and A.S.Pomportsis, "On the Implementation of a Learning Automaton-based Adaptive Wireless Push System", in Proceedings of Symposium on Performance Evaluation of Computer and Telecommunication Systems 2001 (SPECTS'01), July 15-19, 2001, Orlando, USA, pp.484-491.

[11] P. Nicopolitidis, G. I. Papadimitriou and A. S. Pomportsis, "Multiple Antenna Data Broadcasting for Environments with Locality of Demand", IEEE Transactions on Vehicular Technology, vol.56, no.5, pp.2807-2816, September 2007.

[12] K. S. Narendra, M.A.L. Thathachar, "Learning Automata: An Introduction", Prentice-Hall, New Jersey, 1989.

[13] N. Vlajic, C.D. Charalambous and D.Makrakis, "Performance Aspects of data Broadcast in Wireless Networks with User Retrials", IEEE/ACM Transactions on Networking, vol.12, no.4, pp.620-633, August 2004.

[14] C.L.Hu and M.S Chen, "Adaptive Multichannel Data Disemination:Support of Dynamic Traffic Awareness and Push-Pull Time Balance", IEEE Transactions on Vehicular Technology, vol.54, no.2, pp.673-686, March 2005.

[15] Chih-Lin Hu and Ming-Syan Chen, "Adaptive Balanced Hybrid Data Delivery for Multi-Channel Data Broadcast", IEEE International Conference on Communications, 2002, vol.2, pp. 960-964.

[16] M. A. L. Thathachar, P. S. Sastry, "Networks of Learning Automata: Techniques for Online Stochastic Optimization", Kluwer Academic Publishers, 2004.

[17] F. Abtahi, M. R. Meybodi, "Solving Multi-Agent Markov Decision Processes using learning automata", 6th International Symposium on Intelligent Systems and Informatics, 2008, pp. 1-6.

[18] Hong Li, L. Mason, M. Rabbat, "Learning Minimum Delay Paths in Service Overlay Networks", 7th IEEE International Symposium on Network Computing and Applications, 2008, pp. 271 – 274.

[19] E. Ikonen, K. Najim, "Use of learning automata in distributed fuzzy logic processor training", IEE proceedings of control theory and applications, 1997, vol. 144, no.3, pp. 255-262.

[20] S. Ikebou, F.Qian, H.Hirata, "A Parallel Distributed Learning Automaton Computing Model for Function Optimization Problems", Trans. IEE of Japan, vol. 121-C, no.2, Feb., 2001.

[21] K. Schreiner, "Where We At? Mobile Phones Bring GPS to the Masses", IEEE Computer Graphics and Applications, vol.27, issue 3, p. 6-11, May-June 2007.

[22] 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", IEEE Proceedings of the 24th International Conference on Distributed Computing Systems Workshops 2004 (ICDCSW' 04).

[23] Y. Nakajima, H. Shiina, S. Yamane and T. Ishida, "Disaster Evacuation Guide: Using a Massively Multiagent Server and GPS Mobile Phones", IEEE Proceedings of the 2007 International Symposium on Applications and the Internet (SAINT' 07).

[24] R. Matos, D. F. Santos, J. E. Sanguino and A. Rodrigues, “A GPS-based Mobile Coordinated Positioning System for Firefighting Scenarios”, Proceedings of the First Mobile Computing and Wireless Communication International Conference, MCWC2006, p. 209-214.
[25] C. E. Palazzi, “Buddy-Finder: a Proposal for a Novel Entertainment Application for GSM”, IEEE Communications Society Globecom 2004 Workshops, p. 540-543.
[26] K. S. Gilhousen, I.M. Jacobs, R.Padovani, A.J. Viterbi, L. A. Weaver, J. and C. E. Wheatley III, “On the capacity of a Cellular CDMA System”, IEEE Trans. on Vehicular Technology, Vol.40, No.2, May 1991.
[27] C.Y.Lee, “Overview of Cellular CDMA”, IEEE Trans. on Vehicular Technology, Vol.40, No.2, May 1991.