Speeding Up the Adaption Process in Adaptive Wireless Push Systems by Applying Spline Interpolation Technique

V. L. Kakali[1], P. G. Sarigiannidis[2], *Member, IEEE,* G. I. Papadimitriou[3], *Senior Member, IEEE,* and A. S. Pomportsis[4]
Department of Informatics,
Aristotle University of Thessaloniki,
Box 888, 54124, Thessaloniki, Greece
[1] vicky198@csd.auth.gr, [2] sarpan@csd.auth.gr, [3] gp@csd.auth.gr, [4] apompo@csd.auth.gr

Abstract—**In wireless push systems, the server schedules the broadcasts of its information items aiming at satisfying the clients' preferences efficiently. Latest research efforts have proposed adaptive push systems, enhanced with a learning automaton, in which the server has the ability to update its estimated item demand probability vector. This vector indicates the level of the items' desirability. Even though the adaptive push systems are capable of operating in dynamic environments, where the item demand probability distribution changes periodically, the time that the learning automaton needs to adapt its estimated probability vector to a new demand probability distribution leads to degradation of the system's performance. This work addresses this problem, by applying the spline interpolation method to produce an estimation of the changed desirability immediately after this change takes place. A set of indicative feedback samples are collected by the server and the new item demand probability distribution function is approximated, providing the learning automaton with estimated item probabilities, as initial probabilities. Extensive simulation results indicate the superiority of the proposed scheme, in terms of mean response time.**

I. INTRODUCTION

In the area of dissemination of information over asymmetric wireless networks, data broadcasting has appeared as an efficient way of satisfying a large number of clients in a cost efficient way. In applications like, traffic information, weather information, sightseeing guidelines, the broadcast of a single information item will probably satisfy a high number of clients as in such applications the client preferences are often overlapping following a common data demand distribution.

In data broadcasting systems, the server owning a database of information items schedules its broadcasts aiming at serving the clients in the best efficient way. The three major approaches for designing broadcast schedules are the pull, the push and the hybrid one. In the pull (or on-demand) approach [1, 2, 3, 4], the server exploits the client-requests in order to arrange the broadcast schedule. Despite the advantage of these systems in operating to dynamic and continuously altering client demands, they are not scalable for large client populations as client requests will either collide with each other or saturate the server. In the push systems [5, 6, 7], on the other hand, the server is considered to have an a-priori estimation of the client demands and schedules its broadcasts according to these estimates. 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. This weakness is overcome by adaptive push systems [8, 9, 10] that achieve efficient operation in such environments using a learning automaton [11] at the broadcast server in order to provide adaptivity to the overall client population demands and reflect the overall popularity of each data item. Finally, hybrid systems (e.g. [12]) 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 [8].

The system of [8] conserves estimates of the demand probability for each item and uses a learning automaton to adapt its broadcast schedule to the client demands. After each item broadcast, every satisfied client sends to the server a feedback and the server's automaton uses the clients' feedbacks for its adaption procedure.

In this paper, we examine the impact of learning process delay on the operation of the push system. Given that demand probabilities may change as and when new clients subscribe the broadcast service or existing clients unsubscribe or even the region coverage of the system's antenna changes, the learning automaton has to adapt to the new demand probabilities as soon as possible, for the sake of efficiency. The aforementioned adaptive schemes [8, 9, 10] present a major weakness: they infer a serious degradation of the system's performance due to the time that the learning process of the automaton needs to adjust to the new item demand distribution. In order to address this problem, a novel adaptive scheme is proposed that utilizes the spline interpolation technique to speed up the learning process by providing the server with an estimated set of the new demand probability distribution based on the function approximation of the new distribution.

The remainder of the paper is organised as follows: Section II presents the adaptive push framework, Section III presents the novel adaptive scheme, and simulation results are presented in Section IV. Finally, Section V concludes the paper.

II. The Adaptive Push Framework

Learning automata [11] are mechanisms that are able to learn the characteristics of a system's environment. A Learning Automaton is an automaton that improves its performance by interacting with the random environment in which it operates.

In the adaptive push system of [8], the broadcast server is equipped with a learning automaton that contains the server's estimate p_i^{est} of the demand probability p_i for each data item *i* that the server broadcasts, assuming that the number of the data items is equal to M. Apparently, it holds $\sum_{i=1}^{M} p_i = 1$ and $\sum_{i=1}^{M} p_i^{est} = 1$.

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[6]:

$$CF(i) = (T - R(i))^2 \, (p^{est}{}_i / l_i)\,(1 + E(l_i)) / (1 - E(l_i)),\ 1 \le \mathrm{i} \le \mathrm{M} \quad (1)$$

where p_i^{est} 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. Upon the broadcast of item *i* at time *T*, *R(i)* is changed so that *R(i)=T*.

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

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 learning automaton is given by the equation below:

$$p_j^{est}(k+1) = p_j^{est}(k) - L(1 - Fb(k))\,(p_j^{est}(k) - \mathrm{a}),\ \forall \mathrm{j} \ne \mathrm{i}$$
$$p_i^{est}(k+1) = p_i^{est}(k) + L(1 - Fb(k)) \sum_{i \ne j} (p_j^{est}(k) - \mathrm{a}) \quad (2)$$

It holds that $L, \mathrm{a} \in (0,1)$ and $p_i^{est}(k) \in (\mathrm{a},1),\ \forall\, i \in [1,...,M]$, *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. Upon reception of the clients' feedbacks, this number of feedbacks is normalized in the interval of [0,1]. $Fb(k) = 1 - \dfrac{number\ of\ received\ feedbacks}{total\ number\ of\ clients}$ is the system environmental response that is triggered after the server's k^{th} transmission.

III. The Proposed Scheme

A. The System Architecture

The system consists of one base station and a number of *Cl* clients. The base station is equipped with one antenna and a database of *M* different data items. The base station broadcasts the server's data items while clients respond ("vote") to the server's broadcasts.

For the uplink communication, a CDMA coding has been chosen [13, 14]. 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. The CDMA technique for the voting procedure has also been used in [10].

B. Motivation

One of the most important issues regarding the learning automata operation is the speed that the automaton adapts to the dynamic changes of the clients' item preferences. As mentioned on Section II, the learning automaton interacts with the environment and chooses an action (an item to broadcast in our case) based on its item demand probability vector. Using the response of the environment (clients' feedbacks) to a selected item, the automaton updates its item demand probability vector. A new item is then selected according to the updated probability vector and equation 1. If the item demand probability distribution of the automaton's database (estimated item demand probability distribution) is equal or nearly equal to the actual probability distribution (actual item demand probabilities) then it is assumed that the automaton has converged and then it selects the best (or nearly optimal) action. The adaptation speed could be defined as the necessary time that the automaton needs to reach the optimal (or nearly optimal) actions. Eventually, this metric is really important to real environments, since the automaton may take wrong or erroneous decisions until it reaches the environment's item demands, degrading the network performance. So far, approaches for adaptive broadcasting scheduling [8, 9, 10] do not take into account the adaptation speed, when the clients' demands change dynamically and a new actual item demand probability distribution is formed. The main contribution of the proposed scheme is to reduce this negative impact. In other words, this work intends to boost the learning process, giving to the automaton an indicative "image" of the new demand status that has just stated.

In order to evaluate the aforemetioned negative impact, we examined the performance of the previous adaptive push system [8], under successive environment modifications of the actual item demand probabilities. Figure 1 illustrates the indicative findings from our initial simulations. This experiment was conducted with the following assumptions:

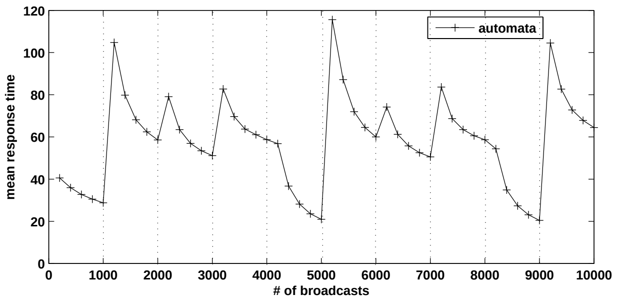

Fig. 1. The evaluation of the adaptive push system of [8] based on indicative simulation results in terms of mean response time.

- The network consists of 3000 clients.
- The available *M* database items are 50.
- The actual item demand probability distribution follows the Zipf distribution. More details about the adopted Zipf equation can be found in Section IV.
- The clients' item demand distribution is modified every 1000 broadcasts. In practice a different value of parameter θ is applied on the Zipf equation, which is produced in a random way.
- The network performance is given by the mean response time metric.

As it is evident from the figure, the network performance is degraded after each environment change (the time that each change happens is noted by the dotted line), causing, at the most cases, high rates of response time. For example, the mean response time is dramatically increased, after the first environment change that takes place at the 1000th broadcast . This has been caused because the actual item demand probability distribution has changed after the 1000th broadcast and the learning automaton still selects items for the following broadcasts based on the previous estimated demand probabilities. Only after the 1420th broadcast the network performance seems to be improved, since the adaptation procedure to the new characteristics of the environment has progressed. Apparently, this modification comes in reality by applying a different value of parameter θ in the Zipf equation. The problem lies exactly at the critical time period that is needed for the learning automaton to update its recorded probability distribution vector and subsequently to be able to take the optimal decisions. Eventually, the delay of the learning automaton's adaptivity reduces the network performance. If a boosting mechanism could be applied to the learning process, the automaton could take more accurate actions regarding the actual client demands in a shorter time. This is the aim of proposed scheme.

C. *The novel adaptive push system*

The target of the proposed scheme is to speed up the learning process of the automaton. In order to achieve this goal, the algorithm utilizes scattered data interpolation with splines. Scattered data interpolation refers to the problem of fitting a smooth surface through a scattered, or non-unifrom, distribution of data samples [15]. The goal of interpolation is to define an underlying function that may be evaluated at any desired set of positions. These positions are formed by the scattered data that the algorithm deliberately acquires from the environment. More specifically, the learning automaton utilizes an update mechanism after each environment change, broadcasting a set of indicative number of items to form a set of scattered positions. Then the curve of the estimated item probability distribution is computed based on the approximate data points with the aim of the spline interpolation technique. The cubic spline interpolation technique has been chosen due to many practical advantages such its minimum curvature property, its high-quality interpolation, simple representation, smoothness etc [16, 17].

In the case that the client population changes the update algorithm is triggered. According to this algorithm, the server broadcasts a set of specific items, called representative items, in a round robin fashion to the clients and collects their feedbacks. Then the cubic spline interpolation is applied to the collected feedbacks and the function that gives the item demand probability distribution is approximated. The set of the representative items, denoted by M^{rep} is a subset of the whole items, hence it holds $M^{rep} \subset M$. The items that form the M^{rep} set are chosen uniformly from the whole set *M* according to the following formula, $M^{rep} = \{iMODf, \, for\, each\, i, i \in M\}$, where f is the frequency factor, which denotes the plurality of the representative samples. Obviously, the factor f specifies the accuracy of the interpolation method, since a set of larger samples leads to more accurate function approximation [18, 19]. For instance, if the M set consists of 100 items and the factor f is equal to 10, then the M^{rep} set consists of 10 items. The server collects the feedbacks, coming from each broadcast and in the next step the cubic spline interpolation is applied in the feedback samples and a function approximation is made based on the M^{rep} samples. The update algorithm is described in Algorithm1.

Algorithm 1: Update Algorithm

```
1. Define the set of items, denoted by M
2. Calculate the set of the representative items, denoted by $M^{rep}$, by applying the following formula: $M^{rep} = \{iMODf, \text{ for each } i,\ i \in M\}$
3. Broadcast the set of $M^{rep}$ representative items in a round robin fashion.
4. Get the feedback of each broadcast, denoted by the $Fb$ set, in the following manner: $fb_1 \text{ for } i_1^{rep}, fb_2 \text{ for } i_2^{rep} \ldots,$ where $fb_1, fb_2 \ldots \in Fb, i^{rep}_1 \text{ and } i_2^{rep} \ldots \in M^{rep}$.
5. Apply the cubic spline interpolation method in the Fb set and produce the approximation function F.
6. Following the approximation function F calculate the estimated item demand probability value for each item.
```

The core broadcast scheduling algorithm is described in Algorithm 2.

Algorithm 2: Broadcast scheduling algorithm

```
1. The server chooses to transmit the item i that maximises the cost function as it has been defined in section II:
   $CF(i) = (T - R(i))^2\,(p^{est}_i / l_i)\,(1+E(l_i))/(1-E(l_i)),\ 1 \le i \le M$
2. Each satisfied client sends to the server its feedback.
3. The learning automaton's estimation probability vector $p^{est}$ is updated, using the received clients' feedbacks $Fb_i$, as it has been defined in section II.
   $p_j^{est}(k+1) = p_j^{est}(k) - L(1-Fb_i(k))\,(p_j^{est}(k) - a),\ \forall j \ne i$
   $p_i^{est}(k+1) = p_i^{est}(k) + L(1-Fb_i(k)) \sum_{i \ne j} (p_j^{est}(k) - a)$
```

IV. The Simulation Environment

The server contains a database of M equally-sized items, with item length l, being equal to the unit. A population of Cl clients is considered. Each client may demand (prefer) a data item with item demand probability equal to p_i. The clients' item demand probability p_i for each item in place i is computed via the Zipf distribution:

$$p_i = c\left(\frac{1}{i}\right)^{\theta}, \quad \text{where } c = \frac{1}{\sum_i \left(\frac{1}{i}\right)^{\theta}}, \quad k \in [1..M], \text{ where}$$

M is the number of database items and θ is the item demand skew coefficient.

The environment changes the item demand probability vector (parameter θ) (a) every 1000 broadcasts and (b) randomly. The number of the representative items of the set of M^{rep} is set equal to 10% of the M set.

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 proposed scheme that uses a combination of cubic spline interpolation and a learning automaton is compared with the system of [8] that uses only the core learning automata mechanism.

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: *Cl=3000*, *Broad=10000*, $\lambda = 0.1$, L=0.15 and a=10^{-4}.

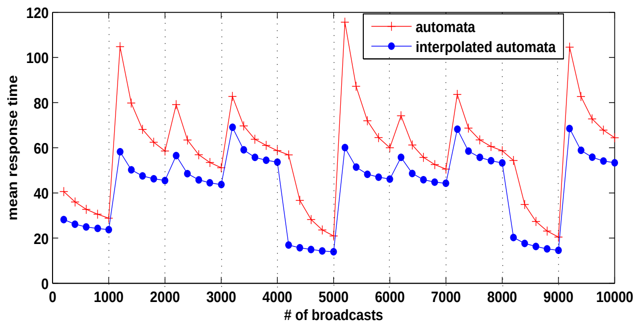

Fig. 2. Mean response time in a system with 50 items. The item demand probability changes every 1000 broadcasts.

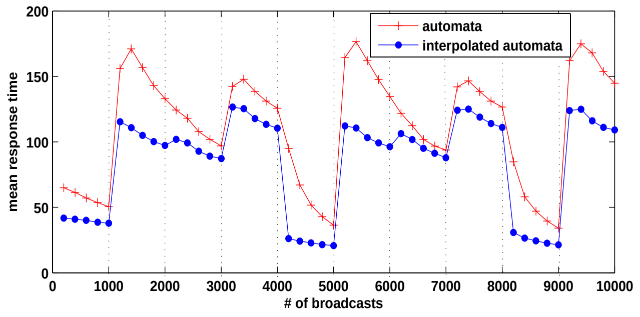

Fig. 3. Mean response time in a system with 100 items. The item demand probability changes every 1000 broadcasts.

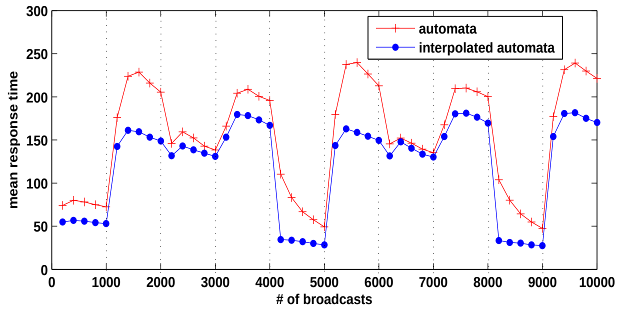

Fig. 4. Mean response time in a system with 150 items. The item demand probability changes every 1000 broadcasts.

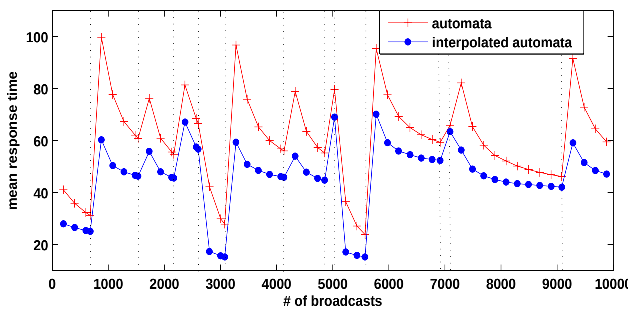

Fig. 5. Mean response time in a system with 50 items. The item demand probability changes randomly.

Figures 2-4 display the mean response time that the two compared schemes achieve for three networks with the number of items M being equal to 50, 100 and 150, respectively. In all of these networks, the item demand probability vector (parameter θ of the item demand Zipf distribution) varies every 1000 server's broadcasts (i.e. 1000th, 2000th, 3000th etc.). After each change of parameter θ, the mean response is calculated every 200 broadcasts or at the time that a new change of parameter θ happens. Upon the change of the item demand probability, the calculation of mean response time restarts without taking into consideration its previous value. This happens in order to take a clear picture of how fast the proposed interpolation-based scheme adapts to the new client demands after each item demand probability change in compare to the scheme of [8] (how fast the estimated probability values in the learning automaton converge to the clients' preferences). Observing the above figures, it is obvious that the proposed scheme achieves lower response time than the one of the core learning automata scheme due to the usage of interpolation method. In other words, after each item demand change, the proposed scheme obtains almost immediately a quite accurate estimation of the new demand probability vector. Then, this estimation is continuously improved through the classic updating probability mechanism of the learning automata operation. On the contrary, the adaptation time of the system of [8] to the environment preferences is much higher (especially when dense item demand changes happens) because the adaptation mechanism is based exclusively on the updating probability mechanism of the learning automata operation. The observed delay of the adaptation process that happens in the core automata scheme infers the low-performed curve, which reflects the higher response time compared to the interpolation-based scheme.

Figures 5-7 depict the performance of the two compared schemes in terms of mean response time, when the item demand probability vector (parameter θ of the item demand Zipf distribution) alters in random time periods (the time units that parameter θ changes is noted with the dotted line). In these three networks, the number of items is set to 50, 100 and 150, respectively. The mean response time is calculated in the same way as it was described above. For example if the first change happens at the 1023rd broadcast and the next at the 1589th, the response time is calculated at 1023rd, 1223rd, 1423rd and 1589th broadcast. These figures, also, reveal the superiority of the suggested scheme, as it achieves lower mean response time during the entire simulation. It is noticeable, that after the 11th change of the item demand probability, the response time of the core learning automata scheme approaches enough the one of the proposed scheme. This is predictable as the time between the 11th and 12th is quite enough for the learning automaton to adapt to the new item demand probabilities. On the other hand, the core learning automaton scheme fails to adapt adequately concerning the frequent changes of parameter θ (e.g. 3rd to 4nd, 4st to 5nd).

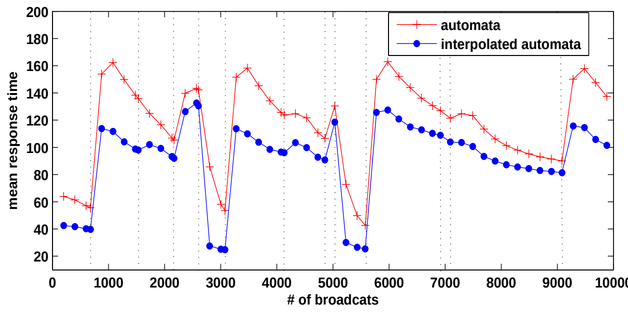

Fig. 6. Mean response time in a system with 100 items. The item demand probability changes randomly.

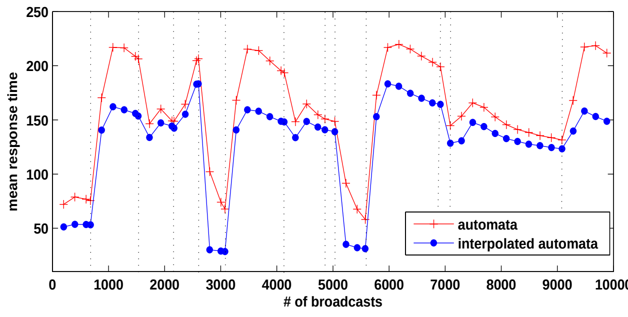

Fig. 7. Mean response time in a system with 150 items. The item demand probability changes randomly.

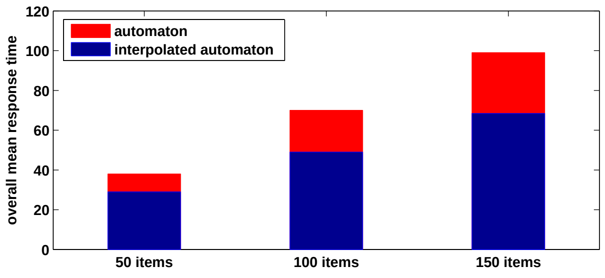

Fig. 8. The overall mean response time of the system where the item demand probability changes every 1000 broadcasts.

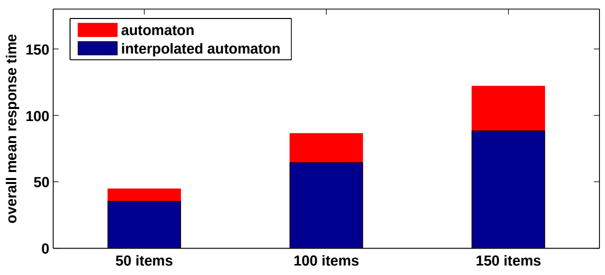

Fig. 9. The overall mean response time of the system where the item demand probability changes randomly.

Finally, figures 8 and 9 indicate that the proposed scheme achieves lower overall mean response time, calculated based on the total performance of the system at the end of the entire simulation procedure.

V. Conclusion And Future Work

A new broadcast scheduling algorithm for wireless push systems was presented in the paper. The examined push system consists of a server and a set of clients. The server broadcasts data items and the satisfied clients respond with a feedback. This process allows the functionality of an adaptive component, realized by a learning automaton. Considering that demand probabilities may change as when new clients subscribe the broadcast service, or existing clients unsubscribe, the learning process adaptation speed may be a negative impact on the system performance. The proposed adaptive algorithm utilizes the spline interpolation technique in order to take a first estimation of the reformed demand probabilities by acquiring indicative feedback samples from the clients. The results of the conducted simulations prove the superiority of the suggested scheme, reducing the mean time that each client waits for a preferred item. Future work will include design of strategies for operating in an environment, in which the time of the client demand changes will be unknown to the server.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] Kakali, V. L., Papadimitriou, G. I., Nicopolitidis, P., Pomportsis, A. S., “A New Class of Wireless Push Systems”, IEEE Trans. On Vehicular Technology, vol. 58, issue 8, pp. 4529-4539, Oct. 2009.

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

[12] 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.

[13] 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.

[14] C.Y.Lee, “Overview of Cellular CDMA”, IEEE Trans. on Vehicular Technology, Vol.40, No.2, May 1991.

[15] S. Lee, G. Wolberg, S. Y. Shin, “Scattered Data Interpolation with Multilevel B-Splines”, IEEE Trans. on Vilualization and Computer Graphics, vol.3, no. 3, July-September 1997.

[16] M. Unser, "Splines: a perfect fit for signal and image processing," IEEE Signal Processing Magazine, vol.16, no.6, pp.22-38, Nov. 1999.

[17] C. Gerald, P. Wheatley, "Applied Numerical Analysis", Addison-Wesley Publishing Company, 2004.

[18] H. Yang, W. Wang, and J. Sun, "Control point adjustment for B-spline curve approximation", Computer-Aided Design, Elsevier, Vol. 36, no. 7, pp.639-652 June 2004.

[19] R. Keys, "Cubic convolution interpolation for digital image processing," IEEE Transactions on Acoustics, Speech and Signal Processing, vol.29, no.6, pp. 1153-1160, Dec 1981.