# An Adaptive Wireless Push System for High-Speed Data Broadcasting

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

***Abstract*— With the increasing popularity of wireless networks and mobile computing, data broadcasting has emerged as an efficient way of delivering data to mobile clients having a high degree of commonality in their demand patterns. This paper proposes a push system that continuously adapts to the demand pattern of the client population in order to reflect the overall popularity of each data item. The adaptation is accomplished using a simple feedback from the clients. We propose that the simple feedback from the clients can be sent at regular time intervals in order to avoid the overhead that would be an incurred for acknowledging all item broadcasts. Simulation results are presented which reveal satisfactory performance in environments where client demands change over time with the nature of these changes being unknown to the broadcast server.**

***Index Terms*— Adaptive Data Broadcasting, Push Systems, Learning Automata.**

## I. Introduction

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

Communications asymmetry is due to a number of facts, such as equipment asymmetry (e.g. lack of client transmission capability, client power limitations), network asymmetry (e.g. small uplink/downlink bandwidth ratio) and application asymmetry (e.g. traffic pattern of client-server applications).

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

This paper enhances the adaptive push-based system of [2] to enable efficient operation in high-speed data broadcasting environments. It suggests the use of a Learning Automaton at the server, which continuously adapts to the demand pattern of the client population in order to reflect the overall popularity of each data item. The adaptation is accomplished using a simple feedback from the clients. Using this approach, information items are transmitted according to client demands, which can be initially unknown to the server and time varying. Contrary to [2], we propose that at regular time intervals the server demands an acknowledgement from those clients that were satisfied by the most recent item broadcast. The smaller frequency for acknowledgements is necessary so as to avoid the overhead that would be incurred when acknowledging all item broadcasts in a high-speed data broadcasting system (e.g. with speeds around 70 Mbps as in 802.16a) spanning an area of radius of several tenths of kilometers.

This acknowledgement has the form of a short feedback pulse. The acknowledging nodes' pulses add at the server that uses the received energy to update the Automaton. The item probabilities estimated by the Automaton converge near the actual overall item demand probabilities of the client population, making this approach attractive for dissemination applications with dynamic demand patterns.

The remainder of this paper is organized as follows: Section II presents the proposed system and discuses how probability updating is performed via client feedback. Section III presents simulation results, which reveal satisfactory performance in environments with dynamic client demand patterns. Finally,

Section IV summarizes and concludes the paper.

## II. The Adaptive Wireless Push System

### A. Providing adaptivity to a push system

Learning Automata [3, 4, 5] are structures that can acquire knowledge regarding the behavior of the environment in which they operate. In the area of data networking Learning Automata have been applied to several problems, including the design of self-adaptive MAC protocols [6, 7, 8, 9].

In the adaptive wireless push system [2], which enhances the non-adaptive one of [10], the server is equipped with an S-model Learning Automaton that contains the server's estimate $p_i$ of the demand probability $d_i$ for each data item $i$ among the set of the items the server broadcasts. The estimation probability vector $p$ stores the server's estimation of the actual demand probability vector $d$ that contains the actual choice probabilities of the various information items averaged over the entire client population. Clearly $\sum_{i=1}^{N} p_i = \sum_{i=1}^{N} d_i = 1$, where $N$ is the number of items in the server's database. At each cycle, the server selects to transmit the item $i$ that maximizes the cost function $G(i) = \left(T - R(i)\right)^2 \frac{p_i}{l_i}\left(\frac{1+E(l_i)}{1-E(l_i)}\right)$, where $T$ is the current time, $R(i)$ the time when item $i$ was last broadcast, $l_i$ is the length of item $i$ and $E(l_i)$ is the probability that an item of length $l_i$ is erroneously received. For items that haven't been previously broadcast, $R$ is initialized to -1. If the maximum value of $G(i)$ is shared by more than one item, the algorithm selects one of them arbitrarily. Upon the broadcast of item $i$ at time $T$, $R(i)$ is changed so that $R(i)=T$.

At regular time intervals (e.g. after several tenths of item broadcasts) the server demands an acknowledgement from those clients that were satisfied by the most recent item broadcast. The aggregate received pulse is then used at the server to update the Automaton. The probability distribution vector $p$ maintained by the Automaton estimates the demand probability $d_i$ (and thus the popularity) of each information item $i$. For the next broadcast, the server chooses which item to transmit by using the updated vector $p$.

When the transmission of an item $i$ does not satisfy any waiting client, the probabilities of the items do not change. However, following a transmission that satisfies clients, the probability of item $i$ is increased. The following Liner Reward-Inaction ($L_{R-I}$) probability updating scheme [3] is employed after the transmission of item $i$ (assuming it is the server's $k^{th}$ transmission):

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

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

Using the probability updating scheme of (1), the item probabilities estimated by the Automaton converge near the actual demand probabilities for each item. This makes this approach attractive for dissemination applications with dynamic client demands. This convergence is schematically shown in Figure 1, which plots the convergence of an item

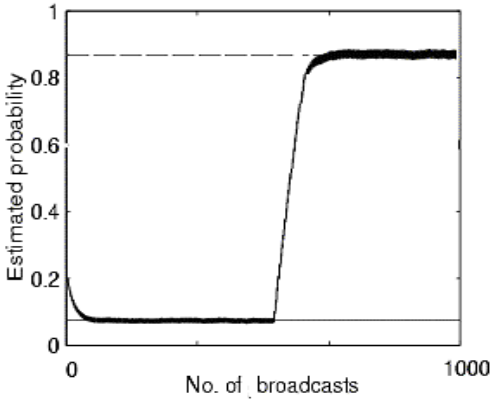


Fig. 1: Convergence of Automaton estimation of the demand probability for a data item.

probability estimate towards the actual overall demand probability for that item in a simulation of a sample scenario with a-piori unknwown client demands that change after some time from about 0.1 to 0.9. It is evident that convergence of the Automaton item probability estimate to the overall client demand for this item is achieved.

The probability updating scheme of Equation (1) is of O($N$) complexity. Therefore the adaptive push method does not increase the computational complexity of the static one The bucketing described in [10] would not further reduce computational complexity in the adaptive method, as complexity would remain of O($N$) due to the probability updating scheme.

The normalization procedure in the calculation of $b$(k) suggests the existence of a procedure to enable the server to possess an estimate of the number of clients under its coverage. This is made possible by broadcasting a control

packet that forces every client in the cell to respond with a power-controlled feedback pulse. The broadcast server will use the power of the received pulses $S$ to estimate the number of clients under its coverage. Then, upon reception of an aggregate feedback pulse of power $Z$ after the server's $k^{th}$ broadcast, $b(k)$ is calculated as Z/S. This estimation process will take place at regular time intervals with the negligible overhead of broadcasting a unit-length item.

However, as the signal strength of each client's pulse at the server suffers a $1/d^n$ type path loss (with a typical $n$=4 [11]), the feedback pulses of clients must be power controlled. To this end, every information item will be broadcast including information regarding the signal strength used for its transmission and acknowledging clients set the power of their feedback pulse to be the inverse of the ratio (signal strength of the received item) / (signal strength of the item transmission). Using this form of power control, the contribution of each client's feedback pulse at the server will be the same regardless of the client's distance from the antenna.

## III. Performance Evaluation

Using simulation, we compared the proposed adaptive approach against the static round-robin broadcast (also known as flat broadcast), which cyclically transmits all items with the same frequency. The flat approach also derives from the method in [10] for equi-probable, fixed-length items and is used here as a measure to count the efficiency of the adaptive approach, since we consider unknown and time varying client demand patterns.

We consider a broadcast server having a database of *Dbs* equally-sized items. The server is initially unaware of the demand for each item, so initially every item has a probability estimate of 1/*Dbs*. Client demands are a-priori unknown to the servers and location dependent. Item broadcasts are subject to reception errors, with unrecoverable errors per-instance of an item occurring according to a Poisson process with rate $\lambda$, as in [10].

We consider *ClNum* clients that have no cache memory, an assumption also made in other similar research (e.g. [10]). Every client accesses items in the interval [1, *Range*], which can be a subset of the items that are broadcast. All items outside this range have a zero demand probability at the client. This item range consists of an integral number of $R$ regions of size *Rsize* items. Items inside the same region are demanded with the same probability of $d(i) = c\left(\frac{1}{i}\right)^{\theta}$ where $c = 1/Rsize\sum_{k}\left(\frac{1}{k}\right)^{\theta}$, $k \in [1..R]$ and $\theta$ is a parameter named access skew coefficient. This is the Zipf distribution used in modeling of client demands in othe papers as well ([2, 10, 12]). For $\theta$=0, the Zipf distribution reduces to the uniform distribution. As the value of $\theta$ increases, the Zipf distribution produces increasingly skewed demand patterns. The Zipf distribution can thus efficiently model applications that are characterized by a certain amount of commonality in client demand patterns.

To simulate some "noise" in client locations, we introduce parameters *Dev* and *Noise*. These parameters determine the percentage of clients that deviate from the initial demand pattern described above and the degree of that deviation respectively. For every client, a coin toss, weighted by *Dev*, is

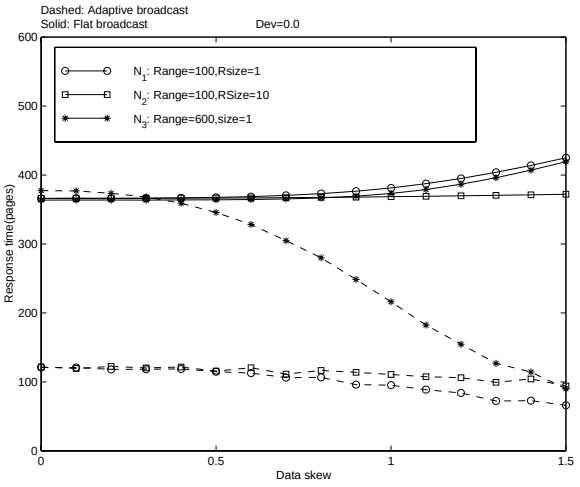


Fig. 2: Overall Mean Access Time versus access skew coefficient $\theta$. Parameter *Dev* is set to 0.

made. If the outcome of the toss states that the client is to deviate from the initial demand pattern, then a new demand pattern for this client is generated. This pattern is produced in the following way: with probability *Noise* the demand probability of each item in the client's demand pattern database is swapped with that of another item that is selected in uniform manner from the interval [1..*DbS*].

To simulate client mobility, each client stays in the same position for time spans, which follow an exponential distribution with, mean $D$ slots. Finally, we set the server to request acknowledgements every *ACKperiod* item broadcasts.

The simulation is carried out until at least $N$ requests are satisfied at each client, meaning that overall, at least *N***ClNum* requests have been served. Afterwards, the demand pattern per client changes again as would happen in a time-varying environment and the simulation continues with the new demand patterns active. This procedure is repeated *Sh* times.

Figures 2, 3 display the results of certain simulation experiments. Those results were obtained with the following parameter values being constant:

- *DbS*=600.
- *ClNum*=10000.
- *N*=2000.
- *Sh*=4.
- *D*=50.
- *ACKperiod*=50.
- *Noise*=0.5.

- $L$=0.15.
- $\alpha$=10-4.
- $\lambda$=0.1.
- *ACKPeriod*=50.

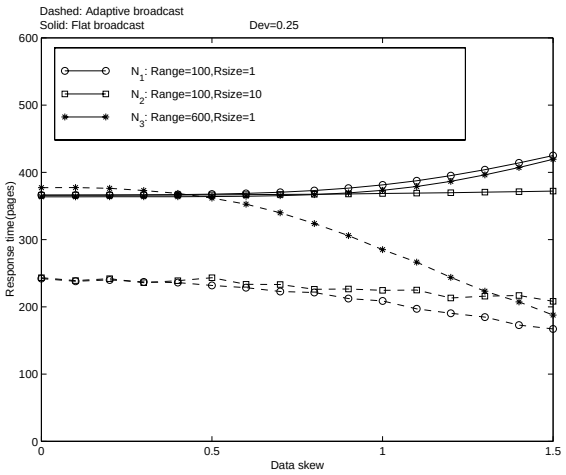


Fig. 3: Overall Mean Access Time versus access skew coefficient $\theta$. Parameter *Dev* is set to 0.25.

Each Figure contains results for a different value of Dev. In each Figure, six plots are presented. These correspond to three pairs of plots that compare the performance of the adaptive approach (dashed plots) to that of the flat one (solid plots). Each pair of plots is characterized by one of the following pair of values:

- Network $N_1$: *Range*=100, *RSize*=1.
- Network $N_1$: *Range*=100, *RSize*=10.
- Network $N_3$: *Range*=600, *RSize*=1.

In Figures 2,3 we observe that the adaptive approach outperforms the flat broadcast in all cases. When all clients follow the same access pattern (*Dev*=0.0) the performance of Networks $N_1$ and $N_2$ is at least three times better than the that of the corresponding flat schemes. This performance improvement logically decreases for increasing values of *Dev*, however, even in the case of *Dev*=0.25 the adaptive schemes are significantly faster than the corresponding flat schemes.

For Networks $N_1$ and $N_2$, even in the case of small values of $\theta$ (which means that the demand pattern is not very skewed) our scheme works best, since it notifies the server to broadcast only the hot-spot of the database more frequently. Thus our scheme would be useful even in cases of applications that access subsets of the server's database with demand patterns close to being uniform (as is the case of $\theta$=0 for Networks $N_1$ and $N_2$ in our plots).

For small values of $\theta$ the performance of $N_3$ is close to that of the corresponding flat scheme. This is due to the fact that in $N_3$ *Range=DbS*, which means that clients pick up items from the entire database of the server. This demand pattern for decreasing values of $\theta$ tends to reduce to a uniform distribution, which leads to the flat broadcast of the server's items. For increasing values of θ however, the performance of $N_3$ increases significantly.

One final note concerns the increasing delay of the flat schemes for large values of $\theta$ in $N_1$ and $N_3$. This is due to the fact that for small and medium values of $\theta$ the demand skew is not very big, which leads clients to select items in a manner close to uniform. For large values of $\theta$ however, the skew is so big that makes the demand probability for all the items in the client's demand pattern practically zero, except for a few ones. Thus, since a client selects only among very few items, it generally has to wait more than half the period of the broadcast to satisfy its requests. The flat scheme in $N_2$ is not subject to this performance increase, since the use of *RSize*=10 produces a less skewed demand pattern.

## IV. Conclusion

This paper proposed a push system that can efficiently operate in high speeds in environments with dynamic client patterns. Adaptation is accomplished using a simple feedback from the clients, which is sent at regular time intervals in order to avoid an overhead at the operation of the protocol. Simulation results are presented which reveal satisfactory performance in environments where client demands change over time with the nature of these changes being unknown to the broadcast server.

## References

[1] P.Nicopolitidis, M.S.Obaidat, G.I.Papadimitriou and A.S.Pomportsis, Wireless Networks, John Wiley and Sons, 2003.

[2] 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.

[3] K.S.Narendra; M.A.L.Thathachar, Learning Automata: An Introduction, Prentice Hall, 1989.

[4] G.I.Papadimitriou, "A New Approach to the Design of Reinforcement Schemes for Learning Automata: Stochastic Estimator Learning Algorithms", *IEEE Transactions on Knowledge and Data Engineering*, Vol.6, No.4, pp. 649-654, August 1994.

[5] G.I.Papadimitriou, "Hierarchical Discretized Pursuit Nonlinear Learning Automata with Rapid Convergence and High Accuracy", *IEEE Transactions on Knowledge and Data Engineering*, Vol.6, No.4, pp.654-659, August 1994.

[6] G.I.Papadimitriou and A.S Pomportsis, "Learning Automata-Based TDMA protocols for Broadcast Communication Systems with Bursty Traffic", *IEEE Communication Letters*, Vol.4, No.3, pp.107-109, March 2000.

[7] G.I.Papadimitriou and A.S.Pomportsis, "Self-Adaptive TDMA Protocols for WDM Star Networks: A Learning-Automata-Based Approach", *IEEE Photonics Technology Letters*, Vol.11, No.10, pp. 1322-1324, October 1999.

[8] G.I.Papadimitriou and D.G.Maritsas, "Learning Automata-Based Receiver Conflict Avoidance Algorithms for WDM Broadcast-and-Select Star Networks", *IEEE/ACM Transactions on Networking*, Vol.4, No.3, pp.407-412, June 1996.

[9] P.Nicopolitidis, G.I.Papadimitriou and A.S.Pomportsis, "Learning-Automata-Based Polling Protocols for Wireless LANs", *IEEE Transactions on Communications*, Vol.51, No.3, pp.453-463, March 2003.

[10] N.H.Vaidya, S.Hameed, "Scheduling Data Broadcast In Asymmetric Communication Environments", *Wireless Networks*, Vol.5, No.3, pp.171-182.

[11] B.Andersen, T.S.Rappaport and S.Yoshida, "Propagation Measurements and Models for Wireless Communication Channels", *IEEE Communications Magazine*, Vol.33, No.1, pp.42-49, January 1995.

[12] S.Acharya, M.Franklin and S.Zdonik, "Dissemination-based Data Delivery Using Broadcast Disks", *IEEE Personal Communications*, Vol.2, No.6, pp. 50-60, December 1995.