On the Performance of Adaptive Wireless Push Systems in High Bit Rate Environments

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

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. The feedback is collected by the server that uses it to update the item probability estimation vector via the Learning Automaton. The item probabilities estimated by the Automaton converge near the actual overall item demand probabilities of the client population, making this approach attractive for dissemination applications with dynamic client demands for data items.

The method to convey the client feedback to the server uses orthogonality in the code domain which, as explained later on, imposes a constraint on the minimum time needed to collect client feedbacks. The latter fact imposes performance limits for the system described in [2] in environments characterized by high server broadcasting speeds, as in such cases, the time period for feedback collection for each item that was broadcast is a performance bottleneck to the system. Thus, contrary to [2], we propose an in-band channel realized over the transmitted data items, over which the server probes the data items in order to estimate their probability of being demanded by the client population. The frequency of probing is selected in such a way so that a continuous flow of broadcasted items is ensured and the performance of the system can scale in proportion to the broadcasting speed of the

978-1-4244-4671-1/09/$25.00 ©2009 IEEE

server.

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 and under various high broadcasting speeds. Finally, Section IV summarizes and concludes the paper.

II. The Proposed Wireless 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 proposed adaptive wireless push system, the server is equipped with an S-model Learning Automaton that contains the server's estimate p_i of the demand probability d_i of the client population, for each data item i among the set of the items the server broadcasts. 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 $G(i) = (T - R(i))^2 \frac{p_i}{l_i}\left(\frac{1+E(l_i)}{1-E(l_i)}\right)$, 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 $R(i)=T$.

As mentioned in the introductory section, the server will probe the data items in order to receive feedback from them so as to estimate the demand probability for each data item. For the feedback transmission, CDMA is chosen [10]. Each client that was waiting for the data item being probed sends its feedback (i.e. one bit) to the base station using a user-specific high-speed code (Long code). At the receiver end (base station), signals are separated by using a correlator, which only accepts signals energy from the specific client's long code and despreads its spectrum. Other co-user signals remain spread because their spreading algorithm is uncorrelated with the desired signal's algorithm and they appear as noise. The CDMA technique has been preferred because CDMA has advantages in several aspects compared to other techniques [11], [12]:

- High capacity. The primary purpose of using CDMA is to achieve high capacity, which in our case translates to a larger number of supported mobile clients. CDMA operates on a common frequency carrier which can result to a frequency reuse factor of one. On the other hand, CDMA can tolerate noise to a great extend due to its wide bandwidth. All these factors contribute to the high capacity of CDMA system [11] which is given by the equation below:

$$P = \frac{W/U}{E_b/N_o} + 1 \qquad (1)$$

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

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

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

The above mentioned requirement will in turn translate to a need for the highest possible duration for the feedback transmission by the clients. However, although benign in low speed broadcasting environments, this requirement will pose a performance bottleneck for the method of [2] when the server

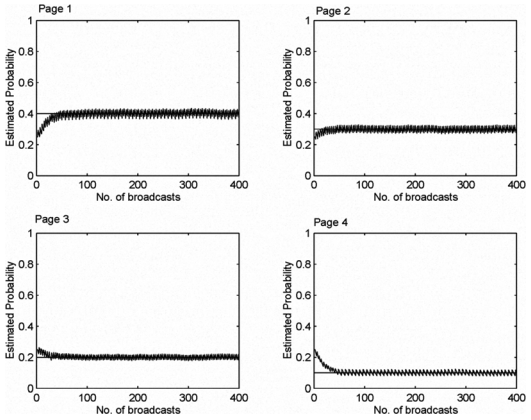

Fig. 1: Convergence of Automaton estimation of the demand probabilities.

broadcasts at high bit rates. This is due to the fact that in such high-speed environments, the ratio of packet transmission time to the feedback transmission time becomes very low, thus the server underutilizes the downlink broadcast channel since it is required to wait for the reception of feedback after each item broadcast. Thus, we propose, that contrary to the method of [2], which a) waits to collect feedback after the transmission of each item by the server, b) states that the collected feedback concerns the data item that was just transmitted, the feedback collection method of the proposed approach is differentiated. Specifically, in the proposed method the feedback a) is not transmitted by the clients after each broadcasted item but rather upon request by the server and b) when requested by the server, the feedback that is received does not necessarily relate to the most recent data item transmission.

To achieve the above assumptions, we realize an in-band channel over the transmitted data items over which the server can request feedback for the item it selects to probe. The in-band channel is realized by piggybacking a $1+\lceil \log_2 N \rceil$ number of bits in the header of each broadcast item i, $1 \le i \le N$, that a) when set to a value representing to decimal j, $1 \le j \le N$, will correspond to the server's request for feedback transmission by the clients that are currently waiting to receive item j, b) when set to zero will indicate that the server does not request feedback after the broadcast of item j.

As far as the details of the probing mechanism are concerned, after the transmission of a probing request, the server will continuously (e.g without probing requests) broadcast items for an aggregate time duration equal to at least the duration of the feedback transmission plus twice the maximum propagation delay from the server to the clients. Both these parameters are known a-priori since the feedback uses one user bit and the user transmission speed U is known. Moreover, the maximum propagation delay is tied to the server transmission range and is also known. Thus, when the time duration since the last probing request exceeds the above-mentioned sum the server will transmit the next item i in the schedule with a piggybacked probe for another item j. In order to select the item to probe, j, we separately apply the cost function $G(j)$ to produce the sequence of probed items by assuming that time elapses only when transmission of items with probing requests are made. This of course entails that the server needs to store a separate vector R' that stores the time when each item was last probed by the server.

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

When the probing of an item j does not yield client feedback due to the fact that no client was waiting for j, the probabilities of the items do not change. However, following a probe for j that yields client feedback the probability of j is increased. The following Liner Reward-Inaction (L_{R-I}) probability updating scheme [3] is employed after the transmission of item j (assuming it is the server's k^{th} probing request):

$$p_m(k+1) = p_m(k) - L\big(1-b(k)\big)\big(p_m(k)-a\big) \quad \forall j \neq m$$
$$p_j(k+1) = p_j(k) + L\big(1-b(k)\big)\sum_{m} \big(p_j(k)-a\big) \tag{2}$$

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

Using the probability updating scheme of (2), 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 page probability estimates versus the overall actual demand probabilities, in a simulation of a sample database comprising four pages and 2000 clients. The client demand pattern is unknown to the server, so

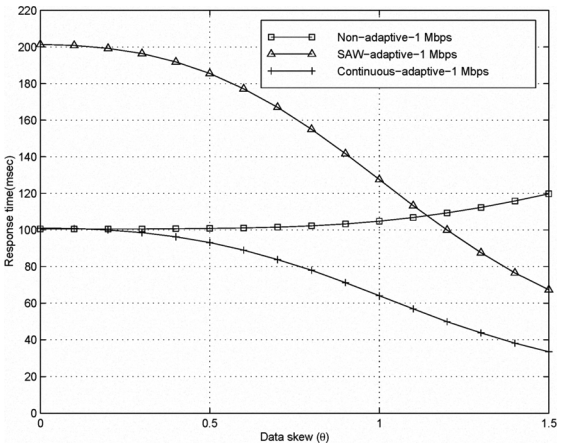

Fig. 2: Overall Mean Access Time versus access skew coefficient θ. Server broadcasts at 1 Mbps.

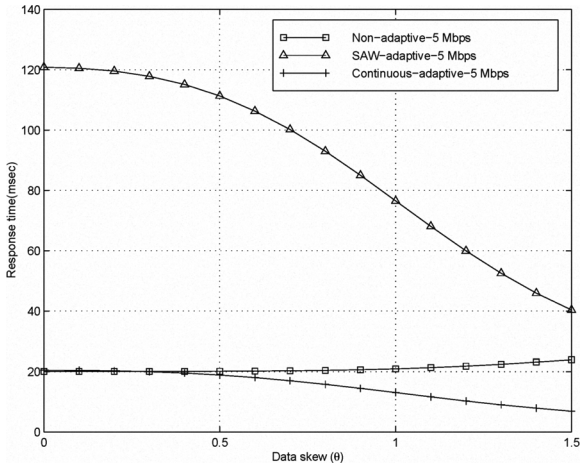

Fig. 3: Overall Mean Access Time versus access skew coefficient θ. Server broadcasts at 5 Mbps.

initially, the four pages have an equal demand probability estimate It is clearly seen, that convergence of the page probabilities estimated by the Automaton to the actual overall demand pattern of the client population is achieved.

The probability updating scheme of Equation (2) is of O(N) complexity. Therefore the adaptive push method does not increase the computational complexity of the static one. The bucketing described in [13] 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 feedback. The broadcast server will use the aggregate received feedback S to estimate the number of clients under its coverage. Then, in subsequent probing requests, upon reception of an aggregate feedback from Z clients after the server's k^{th} probing request, $b(k)$ is calculated as 1-Z/S. This estimation process will take place at regular time intervals with the negligible overhead of broadcasting an item.

III. Performance Evaluation

Using simulation, we compared the proposed system against the one of [2] and 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 [13] for equi-probable, fixed-length items. 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*.

We consider *ClNum* clients that have no cache memory, an assumption also made in other similar research (e.g. [2, 13]). Every client accesses items in the interval [1, *Dbs*]. Thus, every item i, is demanded by the client with a probability of $d(i)=c\left(\frac{1}{i}\right)^{\theta}$ where $c=1/\sum_{k}\left(\frac{1}{k}\right)^{\theta}$, $k\in[1..Dbs]$ and θ is a parameter named access skew coefficient. This is the Zipf distribution used in modeling of client demands in other papers as well ([2, 13, 15]). For θ=0, the Zipf distribution reduces to the uniform distribution. As the value of θ increases, the Zipf distribution produces increasingly skewed demand patterns. The Zipf distribution can thus efficiently model applications that are characterized by a certain amount of commonality in client demand patterns.

The simulation is carried out until at least *NumItems* requests are satisfied at each client, meaning that overall, at least *NumItems*ClNum* requests have been served.

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

- *DbS*=200.
- *ClNum*=2000.
- *NumItems*=2000.
- L=0.15.
- $\alpha=10^{-3}$.
- Item size=10^3 *bits*
- *U=1 Kbps.*
- *W= 15 MHz.*
- E_b/N_o=5.

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

Figures 2, 3, 4 display the results of simulation experiments. Each Figure contains results for a different server broadcasting speed. In each Figure, three plots are presented. The plot termed "Non-adaptive" refers to the round-robin broadcast of the server items, the plot termed "SAW-adaptive" (Stop and Wait adaptive) refers to the method of [2] and the term "Continuous-adaptive" refers to the proposed method.

The main conclusions that can be drawn form Figures 2-4 are the following:

- The method of [2] cannot efficiently utilize high-speed broadcasting servers as its performance is significantly lower (i.e. response time is higher) than that of the other schemes. This is due to the fact that the relatively fixed time that the server needs to wait to collect feedbacks after each item broadcast turns to be a performance bottleneck due to the very small duration of each item transmission time under high broadcasting speeds. For increasing values of θ its performance increases (i.e. response time is lower) as expected [2], however in most cases (e.g. Figs, 3, 4) it is always lower than that of the round-robin scheme.
- The proposed approach efficiently utilizes high-speed broadcasting servers, due to the fact that the proposed method for collecting client feedback does not impose an overhead to item broadcasting. As also expected [2, 13, 15], for increasing values of θ its performance is significantly better than that of the round-robin scheme due to the fact that the learning mechanism efficiently estimates the demand probabilities of each item and uses them in the schedule construction algorithm described in Section 2.
- For small values of θ the performance of the proposed approach is close to that of the flat scheme. This is due to the facts that a) the flat system does not utilize client feedback and is thus not impaired with the afore-mentioned performance bottleneck, b) in the proposed system, small values of θ tend to reduce the demand pattern to a uniform distribution over the server's data items, which in turn leads to the flat broadcast of the server's items. For increasing values of θ however, the performance of the proposed system increases significantly compared to that of the round-robin one.
- One final note concerns the increasing response time of the flat schemes for large values of θ (this is seen more clearly in Fig. 2). This is due to the fact that for small and medium values of θ the demand skew is not very big, which leads clients to select items in a manner close to uniform and corresponding performance close to half the duration of broadcasting the entire database sequentially [15]. For large values of θ 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 [15].

IV. CONCLUSION

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

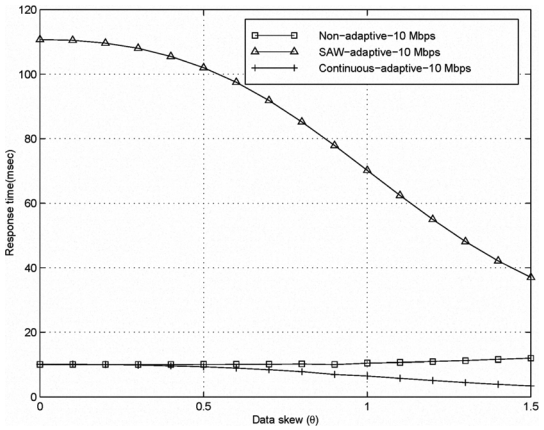

Fig. 4: Overall Mean Access Time versus access skew coefficient θ. Server broadcasts at 10 Mbps.

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] V.Kakali, G.I.Papadimitriou, P.Nicopolitidis and A.S.Pomportsis, "A new Class of Wireless Push Systems", IEEE Transactions on Vehicular Technology, submitted, 2008.

[11] 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 Transactions on Vehicular Technology, Vol.40, No.2, May 1991.

[12] C.Y.Lee, "Overview of Cellular CDMA", IEEE Transactions on Vehicular Technology, Vol.40, No.2, May 1991.
[13] N.H.Vaidya, S.Hameed, "Scheduling Data Broadcast In Asymmetric Communication Environments", *Wireless Networks*, Vol.5, No.3, pp.171-182.
[14] 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.
[15] 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.