Performance Acceleration of Adaptive Wireless Data Broadcasting System for High Data Rate Environments

P. Nicopolitis, G. I. Papadimitriou, A.S.Pomportsis
Department of Informatics,
Aristotle University,
Thessaloniki, Greece.

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 the simple feedback is sent only from clients whose distance from the server does not incur a significant timing overhead for the acknowledgment of an item. Simulation results are presented which reveal satisfactory performance in high-speed environments with a-priori unknown and dynamic client demands.

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 is sent only from clients whose distance from the server does not incur a significant timing overhead for the acknowledgment of an item. Simulation results are presented which reveal satisfactory performance in high-speed environments with a-priori unknown and dynamic client demands.

Keywords— Adaptive Data Broadcasting, Push Systems, Learning Automata.
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. Section III presents simulation results, which reveal satisfactory performance in high-speed environments with a priori unknown and dynamic client demands. Finally, Section IV summarizes and concludes the paper.

II. THE HIGH-SPEED ADAPTIVE 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 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

\[
G(i) = (T - R(i))^2 \frac{p_i}{l_i} - \frac{1}{1 - E(l_i)},
\]

where

\[
R(i) \text{ the time when item } i \text{ was last broadcast, } l_i \text{ is the length of item } i \text{ and } E(l_i) \text{ is the probability that an item of length } l_i \text{ erroneously received. For items that haven't been previously broadcast, } R \text{ is initialized to } -1. \text{ If the maximum value of } G(i) \text{ is shared by more than one item, the algorithm selects one of them arbitrarily.} \]

After each item broadcast 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 (LRI) probability updating scheme [3] is employed after the transmission of item \( i \) (assuming it is the server's \( k^{th} \) transmission):

\[
p_j(k+1) = p_j(k) + L(1-b(k))(p_j(k) - a), \quad \forall j \neq i
\]

\[
p_i(k+1) = p_i(k) + L(1-b(k))\sum_{j \neq i} (p_j(k) - a)
\]

where \( p(k) \) takes values in \((a,1)\) and \( L, a \) take values in \((0,1)\). The role of parameter \( a \) 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 probability estimate towards the actual overall demand probability for that item in a simulation of a sample scenario with a-priori unknown client demands that change after some
time from about 0.1 to 0.55. 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.

However, the above-described system will suffer from a performance decrease in environments where either the coverage area (and thus the maximum propagation delay) is large, or the transmission speed is relatively high. Both these conditions translate into an increased $Max_{Pr\_delay}/Tr\_time$ ratio, where $Max_{Pr\_delay}$ is the maximum one-way propagation delay in the system (thus the propagation delay from the server’s antenna to a client located at the border of the service area) and $Tr\_time$ is the duration of an item’s transmission. The increased $Max_{Pr\_delay}/Tr\_time$ is because the server is able to proceed to the broadcast of item $i+1$ only after a time period $Tr\_Time + 2 * Max_{Pr\_delay} + Feedback\_Time$ has elapsed since the initiation of the broadcast of item $i$, where $Feedback\_Time$ is the transmission duration of the short feedback pulse and is considered very small. Thus it can be seen that when the $Max_{Pr\_delay}/Tr\_time$ ratio is large the server will experience a significant timing overhead for the acknowledgment of each item. To this end, we propose that feedback is sent only from clients located inside a circular area $ACK\_Area$ with the server at the circle’s center. Every other client that is located outside $ACK\_Area$ will never acknowledge an item unless the client moves inside $ACK\_Area$ as explained later. $ACK\_Area$ corresponds to a maximum propagation delay $Pr\_delay$, which is very small, compared to $Tr\_time$.

The server notifies the clients that acknowledgements can be transmitted only by those that are inside $ACK\_Area$ by periodically transmitting via a control item both the radius of $ACK\_Area$ and the power at which it transmits all items. By measuring the power of reception of the control item and using the information regarding the power at which the server transmitted the item, every client can find if its current location is inside $ACK\_Area$ and thus if it will acknowledge data item receptions by feedback pulses.

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 items inside $ACK\_Area$. This is made possible by broadcasting a control packet that forces every client in $ACK\_Area$ 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 in $ACK\_Area$. 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 enables newly-arriving clients in $ACK\_Area$ to acknowledge item receptions after they have responded to the estimation process. It will take place at regular time intervals with the negligible overhead of broadcasting a unit-length item.

As the signal strength of each client’s pulse at the server suffers a $1/d^2$ 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 approach against the adaptive push system of [2]. 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 server. 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 $c = 1/Rsize \sum_{k=1}^{R} \left( \frac{1}{k} \right)^{\theta}$, $k \in [1..R]$ and $\theta$ is a parameter named access skew coefficient. This is the Zipf distribution used in modeling of client demands in other 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 demands, 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 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 $[1..D_{bs}]$.

Client placement takes place on $LP$ different circles, which are equally separated and are located outside the antenna's near field, with the circle at the maximum distance corresponding to the coverage border of the system. The probability of a client to be located on a certain circle is proportional to the perimeter of the circle. Client placement is independent of the client demand patterns.

In the adaptive push system of [2] every client that receives an item, irrespective of the client’s distance from the server, must acknowledge it via a feedback pulse. In the proposed system, we set the server to request acknowledgements only from clients which are inside an $ACK_{Area}$ such that the ratio $Pr_{delay}/Tr_{time}$ is less than $D$, where $Pr_{delay}$ is the maximum propagation delay in the $ACK_{Area}$ from the client to the server antenna and $Tr_{time}$ is the item transmission time. In both systems the duration of the feedback pulse transmission is considered to be very small in comparison to $Tr_{Time}$.

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.

Figures 2-4 display the results of certain simulation experiments. Those results were obtained with the following parameter values being constant:

- $D_{bs}=300$.  
- $ClNum=10000$.  
- $N=1000$.  
- $D=0.1$.  
- $Noise=0.5$.  
- $L=0.15$.  
- $\alpha=10^{-4}$.  
- $\lambda=0.1$.  
- $LP=100$.  
- $Range=300$.  
- $Rsize=1$.  

In each Figure, two plots are presented that compare the performances (Overall Mean Access Times, also known as Response Times) of the proposed adaptive push that utilizes the variable Acknowledging area and the adaptive push one of [2] for various values of the ratio $Max_{Pr_{delay}}/Tr_{time}$. Overall Mean Access Time is defined as the mean waiting time among the client population for an item. Each pair of plots is characterized by one of the following pair of values:

- Network N1: $\theta=0.5$, $Dev=0.0$.  
- Network N2: $\theta=1.0$, $Dev=0.0$.  

![Fig. 2: Response Time versus $Max_{Pr_{delay}}/Tr_{time}$ ratio. Dev= 0, $\theta=0.5$.](image1)

![Fig. 3: Response Time versus $Max_{Pr_{delay}}/Tr_{time}$ ratio. Dev= 0, $\theta=1.5$.](image2)

![Fig. 4: Response Time versus $Max_{Pr_{delay}}/Tr_{time}$ ratio. Dev= 0.2, $\theta=1.0$.](image3)
• Network N2; \(\theta=1.5, \Dev=0.0\).
• Network N3; \(\theta=1.0, \Dev=0.2\).

In Figures 2-4 we observe that the performance of the adaptive push system of [2] worsens (the response time increases) as the ratio \(\text{Max}_\text{Pr} \_{\text{delay}}/ \text{Tr} \_\text{time}\) increases. This is due to the fact that an increase of the propagation delay compared to the item transmission time means that for each item transmission, the server of the adaptive push system of [2] must wait a relatively longer proportion of the item transmission time so as to collect the pulses of every acknowledging client in the entire system service area. This translates to an increased overhead per item transmission, which is seen that significantly degrades the performance of the adaptive push system of [2] when \(\text{Max}_\text{Pr} \_\text{delay}\) approaches \(\text{Tr} \_\text{time}\).

The above-mentioned deficiency for the adaptive push system of [2] is not observed in the proposed push system. This is due to the fact that in the proposed system the server will wait to receive feedback only from those clients located at such distances so that the ratio \(\text{Pr} \_\text{delay}/ \text{Tr} \_\text{time}\) is less than 0.1, where \(\text{Pr} \_\text{delay}\) is the maximum propagation delay from such a client to the server. This of course translates into a fixed time overhead for the acknowledgement of each item transmission. This fixed overhead, which is independent of the \(\text{Max}_\text{Pr} \_\text{delay}/ \text{Tr} \_\text{time}\) ratio, is the reason that the performance of the proposed system is stable and does not depend on the value of the \(\text{Max}_\text{Pr} \_\text{delay}/ \text{Tr} \_\text{time}\) ratio. Furthermore, the performance improvement over the adaptive push system of [2] becomes significant when \(\text{Max}_\text{Pr} \_\text{delay}\) becomes comparable to \(\text{Tr} \_\text{time}\) (e.g. 100%-200% better when \(\text{Max}_\text{Pr} \_\text{delay}/ \text{Tr} \_\text{time}=1\)).

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. The simple feedback is sent only from clients whose distance from the server does not incur a significant timing overhead for the acknowledgment of an item. Simulation results are presented which reveal satisfactory performance in high-speed environments with a-priori unknown and dynamic client demands.

REFERENCES