A Novel Push-and-Pull Hybrid Data Broadcast Scheme for Wireless Information Networks

Jian-Hao Hu, Kwan L. Yeung, Gang Feng and K.F. Leung
Department of Electronic Engineering
City University of Hong Kong
Hong Kong, PRC
Tel: (852) 2788-7730 Fax: (852) 2788-7791
E-mail: {jhhu, kyeung, gfeng}@ee.cityu.edu.hk

Abstract: *A new push-and-pull hybrid data broadcast scheme is proposed for providing wireless information services to three types of clients, general, pull and priority clients. Only pull and priority clients have the back channel for sending requests to the broadcast server. There is no scalability problem with the hybrid scheme because the amount of pull and priority clients is very small. Based on the requests collected from pull and priority clients, the server estimates the interest pattern changes of the whole client population. Then the broadcast schedule on the push channel for the next broadcast cycle is adjusted. Besides push channel, a small amount of broadcast bandwidth is allocated to a pull channel. The data to be broadcast on the pull channel is decided by the server in real-time and priority is given to requests from priority clients. Simulations show that with a time-varying client interest pattern, the average data access time for all three types of clients can be minimized. Because of the priority in using the pull channel, priority clients can achieve the lowest access time and pull clients can achieve a lower access time than general clients. To further improve the performance, the hybrid scheme with local client cache is also investigated.*

1. Introduction

The advances in Internet and wireless networks have fueled the development of a wide range of data dissemination based applications. These applications involve information feeds (e.g., stock, weather forecast), traffic information, electronic personalized newspaper, software distribution and entertainment delivery. Using the traditional client-server unicast approach is not effective for data dissemination applications because these applications are tremendous in scale, have a high-degree of overlap in client interests, and are asymmetrical communications (heavy traffic from the information server to clients). In wireless networks, data broadcast is independent of the client population and thus it plays an important role in dissemination based applications. Owing to the high-degree of overlap in client interests, data broadcast can achieve high bandwidth utilization and fast data response time. There are two basic architectures for a data broadcast system, push-based broadcast and pull-based broadcast [6,10,11,14].

From the clients' viewpoint, an efficient data broadcast scheme should have a fast data access time (or response time) and a low mobile power consumption (due to small/light-weighted batteries used in client terminals). The data access time is the time elapsed from the moment that a client makes a query to the moment that the requested data item is fetched/received. For time-critical applications such as on-line stock price service system, the access time should be guaranteed. To reduce the inconvenience caused, an efficient data broadcast scheme should also minimize the power consumption of mobile terminals.

The pull-based algorithm proposed in [1] cannot meet the large-scale population requirement. The RXW algorithm developed in [3] can provide excellent average access time performance for a large-scale system, but it cannot guarantee the access time for *individual* classes of clients, and thus it is not suitable for clients with time-critical applications. Several index tree algorithms are studied in [1,2]. They can give a near optimal performance for push-based systems. But the excellent performances are obtained based on the assumption that the system can correctly estimate the interest pattern of all clients. In order to reduce mobile terminal power consumption, efficient index and signature schemes are proposed in [7,9,13]. A novel balancing push and pull for data broadcast is proposed in [8]. The results in [8] show that a client back channel can provide significant performance improvement in the broadcast environment, but unconstrained use of the back channel can result in scalability problems due to server saturation/overload. This balancing broadcasting system again cannot support clients with time-critical applications since it provides no access time guarantee for such services. In order to get low access time and power consumption, mobile caching scheme is studied in [4,5]. This is because if the requested data can be found at the local cache, the access time will be minimized. Recently a fault-tolerant broadcast scheme is proposed in [12] to provide error-free data transmission in wireless channel. These techniques are useful for data broadcast system realization, but they cannot solve all the problems in push-based or pull-based systems addressed above.

In this paper, a new data broadcast scheme is proposed with all the requirements for an efficient broadcasting system in mind. We call it *push-and-pull hybrid data broadcast*, or hybrid scheme in short. Unlike existing schemes in the literature, three classes of clients with different access time requirements are supported. They are general clients, pull clients and priority clients, where the priority clients have the highest priority and the general clients have the lowest. Using the hybrid scheme, the system can provide a very short data access time for priority clients and thus time-critical applications can be efficiently supported. In the next section, the hybrid scheme is described in details. In Section 3, we present the broadcast server model and the client model for simulations in Section 4. To study the time-varying client interest pattern, a new model that captures both continuous variation and burst variation in clients' interest pattern is proposed. In Section 4, simulation results show that the hybrid scheme is very effective in reducing the data access time for each class of clients. Finally, we conclude the paper in Section 5 by highlighting some further possible research topics in this area.

2. Push-and-Pull Hybrid Data Broadcast Scheme

2.1 Broadcast Cycle Structure

In a data broadcast system, the server periodically broadcasts all data items in the database to the clients via the wireless channel. A broadcast cycle is defined as the interval that every data item in the database has been pushed out (i.e. broadcast) for *at least* once. The time required for broadcast one data item is called a data slot. Using the hybrid data broadcast scheme, each broadcast cycle consists of *interleaved* push data slots and pull data slots as shown in Fig. 1. The push slots are organized into many equal-sized segments. At the end of each segment, a pull slot is attached. The data items broadcast

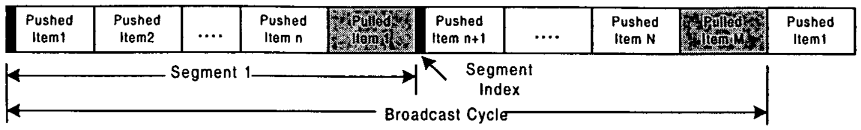

Fig.1. The frame structure of a broadcast cycle.

on the push slots, or *pushed data items*, in the current cycle have been *pre*-scheduled by the server in the previous broadcast cycle. The scheduling of pushed data items is based on the collected client request statistics from the back channel (to be described in more details later on). The data items broadcast on the pull slots, or *pulled data items*, are scheduled in *real-time*. That is, if the next broadcast slot is a pull slot, the server selects a data item to broadcast based on the requests received up to the current slot. It should be noticed that the bandwidth assigned to pushed data is much larger than that of the pulled data.

In this paper, we consider a broadcast system that supports three classes of clients: general, pull and priorty clients. All clients can listen to server broadcasting on both push and pull slots. General clients are low power consumption terminals with the least stringent data access time requirement. They cannot access the back channel for sending requests to the server. For pull and priority clients, the server can schedule the requested data item for broadcasting using pull slots based on the requests received from the back channel. Thus pull and priority clients can get a better access time performance. Priority clients are targeted for users with time-critical applications such as frequently updated stock market price services. They have the most stringent requirement on data access time. Therefore data items requested by a priority client will be broadcast over the pull slots with a higher priority than pull clients. Without loss of generality, we assume that the number of general clients in the system is far more than that of the pull clients, and the number of pull clients is more than that of the priority clients. It should be noted that there is no scalability problem using the hybrid scheme, as the system only allows a very small client group to have the capability for requesting data items.

The segment index shown in Fig. 1 contains the schedule information of which data item is to be broadcast in the push slots of the current segment. A mobile only needs to wake up and listens to the segment index and then sleeps for the rest of the time until the required data item is broadcast. This helps to minimize the client's active time and thus reduce its power consumption. For the remaining discussion in this paper, we only focus on the data access time performance while assuming this index is always attached without explicitly mentioning it again.

2.2 Broadcast Disk Scheduling

Based on the clients' interest pattern, we assign data items into different groups (or, disks) with different broadcasting frequencies. The data items on the popular/hot disk are broadcast with a higher frequency. Fig. 2 shows an example of 10 data items in a database.

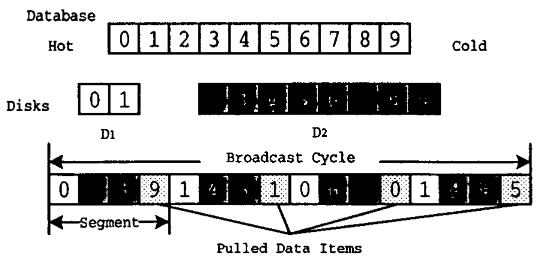

Fig.2. Broadcast schedule of the hybrid scheme.

They are ranked such that data item 0 is most popular (hot) and data item 9 is least popular (cold). We group them into two disks, where items 0 and 1 on disk D_1 and the remaining items on disk D_2. Let the relative disk broadcast frequency ratio be 2:1. The resulting broadcast cycle consists of 4 segments and 16 broadcast slots. Each segment has 4 slots, where the first 3 slots are for pre-scheduled pushed data items and the last slot is for pulled data item. Focusing only on the push slots in a broadcast cycle, data items on D_1 are broacast twice while those on D_2 are broadcast once. The data items broadcast in the pull slots in Fig. 3 are selected by the scheduling algorithm *RPW* to be described in Section 2.3.

Assume that there are N data items with access probabilities λ_1, $\lambda_2,\ldots,\lambda_N$ respectively. If their broadcast frequencies are $f_1, f_2,\ldots, f_N$, then the expected data access time for a pure push-based scheme is:

$$T_{ac} = \frac{T_B}{2}\sum_{i=1}^{N}\frac{\lambda_i}{f_i}$$

where T_B is the broadcast cycle length.

In practice, the clients' interest pattern is time-dependent and is very difficult to correctly estimate. A good broadcast scheduling should therefore adapt to the clients' interest pattern variation. In the proposed push-and-pull hybrid scheduling scheme, we use the request statistics received from the pull clients in the current estimation window (e.g. current broadcast cycle) to estimate the access probabilities of the data items in the next window. Then the broadcast schedule on the push channel in the next window is adjusted by swapping the currently-most-popular data items in the cold disk with the currently-less-popular items in the hot disk. Of course, estimation noises/errors exist. As we are going to show in Section 4, the proposed hybrid scheme is robust to estimation noises. To smooth out the instantaneous variations in clients' data access pattern, a larger window size for collecting requests from pull clients is preferred. The tradeoff is that the system becomes less responsive to changes.

2.3 *RPW* Scheduling for Pull Channel

Next we describe how the requests from both pull and priority clients are processed by the server. We modify the method used in [3], which does not consider multiple client classes. We call the new method *RPW* algorithm. Using the *RPW* algorithm, the server maintains three tables: Request table, Priority table and Waiting-time table, for all data items in the database. The request table records the number of requests for each data item. The priority table records the aggregated priority of the clients requested for each data item. The waiting-time table records the waiting time of the first (unsatisfied) request for each data item. If one data item is broadcast (using either push or pull data slots), the corresponding entries in all *three* tables will be cleared/reset. When a pull slot arrives, the server selects the item with the maximum *RPW* value to broadcast, where *RPW* is given by

$$RPW = (R \times W)^{P+1},$$

where R is the number of requests for an item, W is the waiting time since the first request for this item arrives, P is the sum of the priorities of all the clients who have requested for this data item. In this paper, we have assumed P=0 for a pull client and P=1 for a priority client. Since the primary reason of using P as an exponent in the above expression is to give priority to priority clients, the exact

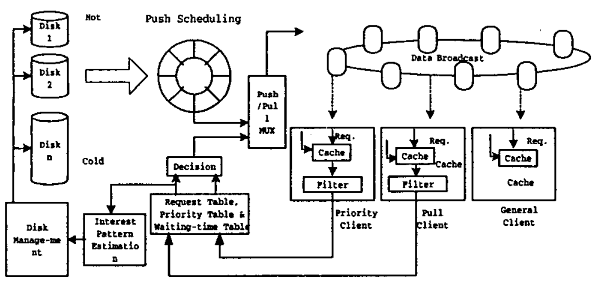

Fig.3. Data broadcast model.

value of *P* is not crucial as long as it can distinguish the priority among different classes of clients.

The basic idea of *RPW* algorithm is to select the data item which has the largest number of requests, with the longest request waiting time, and with the highest agregated priority. Since the broadcast period is divided into small segments and the priority of pull client is zero. The request from priority clients is responded within a very short time interval and thus is good for time-critical applications. When priority traffic is heavy, the pull channel will be dominated by the traffic from priority clients. The requests from pull clients can then be serviced by the push data slots, which are scheduled based on the estimation of clients' interest pattern. When the priority traffic is low, the pull channel can be used by pull clients to improve their access time performance. Therefore, the access time performance of pull clients is less than that of the priority clients but is better than general clients.

2.4 Cache Replacement Algorithm at Mobile Terminal

We consider using cache at each mobile terminal to reduce the access time and mobile power consumption in this section. Since the pushed data broadcast is scheduled based on the client interest pattern, popular data items can be easily identified and stored at the local cache for achieving a higher cache hit probability. However, if the server's broadcast schedule is poorly matched with the client interest pattern, the use of the cache usually can not lead to obvious performance improvement.

Several cache replacement algorithms have been proposed in the literature. In this paper, we adopt the *PIX* cache replacement algorithm [14]. Using the *PIX* algorithm, a client keeps track of its access probability for each data item (denoted by *P*) using its access history. The broadcast frequency of each data (denoted by *X*), obtained by listening to the push channel, is also recorded. When a new data item is received and the cache is full, its *P/X* ratio is compared with the *P/X* ratios of all cached data items. The one that gives the lowest *P/X* ratio will be replaced from the client's cache. For more details, please refer to [14]. As we are going to show in Section 4, *PIX* gives an excellent performance using the proposed hybrid data broadcast scheme.

3. Modeling Data Broadcast System

The data broadcast simulation model used for studying the hybrid scheme is shown in Fig. 3. The priority clients and pull clients filter every request by checking with its cache (if available) before submitting the request to the server using the back channel. The server broadcasts a data item either according to the pre-determined broadcast schedule using push slots, or as a response to the request of the pull and the priority clients using pull slots. The selected data items are then merged together by the *Push/Pull* multiplexer before sending/writing onto the broadcast disk following the broadcast cycle structure shown in Fig. 1.

3.1 Client Model

The parameters that describe the operation of the three classes of clients are summarized in Table 1.

Table 1 Client Parameter Description.

CacheSize	Client cache size (in terms of data items it can store)
$\lambda_{pulling}$	Request arrival rate of the pull clients
$\lambda_{priority}$	Request arrival rate of the priority clients
Noise	% Error in estimation of interest pattern deviation for general clients
θ	Zipf distribution parameter (used for model clients' interest pattern)

Now let us consider the aggregated stream of requests generated by the whole client population (including that from the general clients although their requests can not reach the server). Let the aggregated request arrival process follow a Poisson process with mean λ requests per broadcast slot. Let a client request for data item *i* with probability b_i, where $\sum_{i=1}^{N} b_i = 1$. Hence the requests for item *i* are generated according to a Poisson process with rate $\lambda_i = b_i\lambda$. Let the data item access probabilities for all clients obey the Zipf probability distribution, such that

$$b_i = \frac{i^\theta - (i-1)^\theta}{N^\theta}.$$

The Zipf distribution (with parameter θ) is frequently used to model the skewed (i.e., non-uniform) interest patterns.

In the proposed hybrid scheme, we use the statistics of the pull clients' requests collected from the back channel to estimate the interest pattern of all clients in the system. Although the clients have high-degree of interests overlapped, there are noises/errors in the estimation. The effect of estimation noises to the scheduling scheme performance is studied in Section 4. Since we have assumed that the priority clients have higher priority using the pull slots. The request arrival rate of priority clients $\lambda_{priority}$ will affect the data access time performance of the pull clients. Simulation details are referred to Section 4.

3.2 Server Model

The parameters that describe the operation of the server are summarized in Table 2. We assume all data items are of equal size. Time on the broadcast channel is divided into slots where each slot can accommodate one data item. The server broadcasts data items in the range of 1 to *DBSize* using the proposed hybrid scheme in Section 2.

Table 2 Server Parameter Description

DBSize	Number of data items in database.
NumDisk	Number of disks.
$DiskSize_i$	Size of disk *i* (i.e. the number of data items it can store).
$RelFreq_i$	Relative broadcast frequency of disk *i*.
PullRatio	Ratio between the pushed data slots to pulled data slots.
SegLgh	The length of a broadcast segment.
Variation_Spd	The slow variation speed of the clients' interest pattern.
Burst_Variation	The burst variation (in data items) of the clients' interest pattern.
Resch_Wnd	The window width for re-scheduling a broadcast cycle.
Shift(t)	The amount of shift (in data items) in clients' interest pattern at time *t*.

According to the estimation of clients' interest pattern, we assign the hottest data items to disk #1, the next hottest items to disk #2, and so on so forth (as shown in Fig. 3). The clients' interest pattern is time-dependent and is impossible to have an exact model for it. In this paper, we argue that for a practical system, the variation of the clients' interest pattern consists of two components: *slow speed shift in interest pattern* and *occasional burst variation.*

We use *Burst_Variation*, *Variation_Spd* and *Shift* to model the variation of the clients' interest pattern. The relationships among these parameters are summarized by the following equation:

$$Shift\ (t) = \bmod\left(\left[\int_0^t Variation\_Spd\ (x)dx\right] + Burst\_Variation\ (t),\ N\right) + Noise\ * DBSize \quad (1)$$

where $[x]=k$, for $k\text{-}1<x\leq k$. It is the accumulated *slow* shift in clients' interest pattern from time 0 (i.e. system starts) to time *t*. *Noise* represents the percentage error in estimating the shift in clients' interest pattern. Consider a database with 1000 data items, a noise level of 20% means that the server inaccurately estimates the access probability of data item 1 to be that of item 201, item 2 to be item 202, and so on so forth.

If $Shift(t) = \Delta$ at a given time *t*, the access probability for data item *i* will be:

$$b_i' = b_k$$

$$k = i + \Delta \text{ for } i = 1, 2, ..., N - \Delta;$$

$$k = i + \Delta - N \text{ for } i = N - \Delta +, ..., N$$

where b_k is the access probability for data item *k* at the moment when the system starts to broadcast. It should be noted that this model only captures the *shift* in the clients' interest pattern.

The server records the request information of the pull clients during the *Resch_Wnd*, and re-arrange the data items in the disks. Then the server will clear the old estimation information and start to get new information in the next *Resch_Wnd*. In order to have a responsive system that is free of system oscillation, we set *Resch_Wnd* to one broadcast cycle, the smallest possible duration. After each broadcast cycle, *only* a small portion of the data items in the hot disk will be swapped with the items in the cold disk. In Section 4, we limit up to 2 data items can be swapped in and out in each broadcast cycle.

4. Performance Evaluations

In this section, we use simulations to study the performance of the hybrid scheme in data broadcast with time-varying client interest patterns. The primary performance metric is the data access time in broadcast slots. The server database size is 1000 items and a two-disk broadcast is used. The size of the fast disk is 200 items, and the slow disk is 800 items. The relative broadcast frequencies of the two disks are 2:1. Two *PullRatios*, 10:1 and 10:2, are used. For *PullRatio* = 10:1, each segment has 10 push data slots and one pull data slot. Then the total broadcast cycle consists of 1760 slots. Similarly, for *PullRatio* = 10:2, each segment has 10 push slots and 2 pull slots. The total broadcast cycle length becomes 1820 slots. The *Resch_Wnd* is set to one broadcast cycle, i.e. 1760 slots for 10:1 and 1820 slots for 10:2. The estimation noise/error is set to three different levels, 0%, 10% and 20%. The performance of both pure push-based scheme, i.e. all clients can only listen, and pure pull-based scheme, i.e. all clients can send requests via the back channel, are studied for comparison.

First we consider the case that the clients' interest pattern has a slow continuous shift over the time, i.e. we set the *Burst_Variation* to 0. The interest pattern variation speed, *Variation_Spd*, is set to 0.0003 items/slot. The request arrival rates from pull clients and priority clients are fixed at 10 items/slot and 0.01 items/slot. Fig. 4 shows the data access time of the three classes of clients against time, or the number of broadcast cycles. From Fig. 4, we can see that the data access time using the hybrid scheme remains steady for all three classes of clients, while that of using pure push-based scheme increases. We can also see that as the *PullRatio* changes from 10:1 to

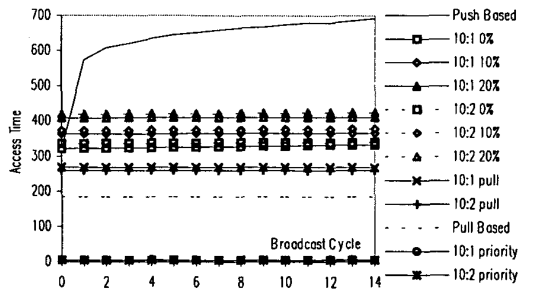

Fig. 4 Data access time performance with slowly varying client interest pattern.

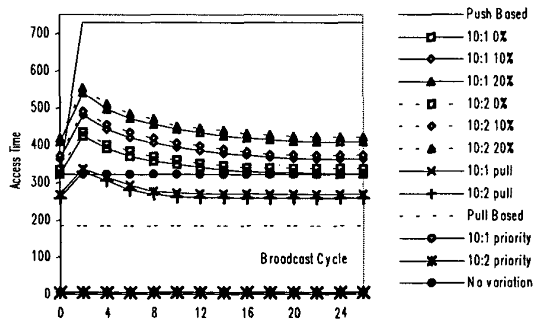

Fig. 5 Data access time porformance with burst variation in client interest pattern.

10:2, the access time performance of the general clients slightly increases. The data access times of the priority clients are 5.3 and 4.6 slots for *PullRatios*=10:1 and 10:2. This represents a significant performance improvement over pull clients, general clients, as well as clients using either pure pull-based or pure push-based schemes. Therefore, the short access time allows the priority clients to use time-critical applications.

Fig. 5 shows the access time performance of the three types of clients with burst variation in clients' interest pattern. In this case, we set *Variation_Spd* to 0, and *Burst_Variation* equal to *100* data items. We let burst variation occur at the end of the first broadcast cycle. From Fig. 5 we can see that the access time of using pure push-based scheme increases from 318.4 to 728.9 at the second broadcast cycle and remains at that level. The access time of the general and pull clients using the hybrid scheme increases slightly and then gradually reduces to the level before the burst variation occurs. This is due to the broadcast schedule is re-calculated based on the interest pattern estimation. Since priority clients dominate the use of the pull channel, their access time is not significantly affected by the burst variation as shown in Fig. 5.

Next we focus on the data access time performance for both pull clients and priority clients under different relative traffic loads. For general clients, its data access time is in general not (or, insignificantly) affected by the relative amount of traffic from pull and priority clients. In this simulation, we set $\lambda_{priority} = \lambda_{pulling}/10$. From Fig. 6, we can see that the access time of the priority clients increases as the $\lambda_{priority}$ increases. When $\lambda_{priority}$ is large, the priority clients have to wait for pull channel. Therefore, the access time of priority clients is worse than that of using pure pull-based scheme. The access time of pull clients increases slightly, from 260.3 to 272.4 slots for *PullRatio* increases from 10:2 to 10:1. From Fig. 6, we can conclude that the population of the priority clients should be limited in order to get a good system performance for priority clients, or the system should assign more bandwidth to the pull channel when the priority clients' request is heavy. Generally speaking, if $\lambda_{priority}$ is less than one item per segment, the performance of the priority clients will be guaranteed.

Now we study the effect of using client cache for pull clients *only*. We set the pull client cache sizes to *CacheSize* = 50 and 100 respectively. The request arrival rate of the pull clients ranges from 1 to 120 items/slot, and $\lambda_{priority}$ equals to 0.01 items/slot. Fig. 7 shows the cache hit probability against the pull clients' request arrival rate. Again the performance of using the pure pull-based scheme is plotted for comparison. The cache replacement algorithm used by pure pull-based scheme is the *Least Recently Used* (LRU) algorithm. Using LRU, the cached data item that has not been used for the longest time will have the highest priority to be replaced. For the proposed hybrid data scheduling scheme, the *PIX* algorithm is used. From Fig. 7, we can see that with the *PIX* algorithm, the cache hit probability of the hybrid scheme is higher than that of the pure pull-based scheme. Due to the high cache hit rate, the hybrid scheme can therefore achieve a better access time performance.

5 Conclusions

A novel push-and-pull hybrid scheme for data broadcast in wireless information networks was proposed in this paper. Unlike conventional approaches, three classes of clients are supported, general, pull and priority clients. The pull clients and the priority clients can send requests to the server via a back channel and get service from the pull channel, where the pull channel only occupies a very small part of the whole broadcast bandwidth. The server estimates the clients' interest pattern variation using the request statistics collected from the back channel. Then it adjusts the push channel broadcast schedule for the next broadcast cycle. The simulation results showed that the hybrid scheme is robust to the interest pattern variation as well as the estimation errors. Since the priority clients have priority in using the pull channel in the broadcast bandwidth, the average data access time for the priority clients is the shortest. On the other hand, since pull clients can send requests via the back channel, their average access time is shorter than that of the general clients. Since the majority of the client population is general clients, the hybrid scheme does not face the scalability problem.

We have also studied the effect of using local cache at each client terminal. An efficient cache replacement algorithm called *PIX* algorithm was evaluated together with the hybrid scheme. Because of the robust nature of the hybrid scheme, we showed that the *PIX* algorithm has an excellent performance with a cache hit probability about 0.6 to 0.7 in our simulations.

In this paper, we have limited our scope to broadcasting *public* information that is interested by many clients. Another interesting area for further research is to allow some pull clients to access *personalized* information (e.g. emails) using the broadcast pull channel at the expense of higher tariff. How to efficiently arrange the public information and the personalized information on the shared broadcast bandwidth is a problem for future research.

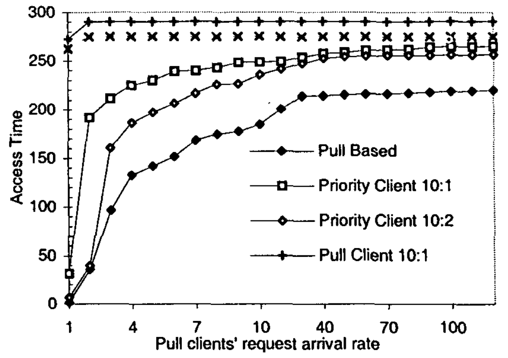

Fig. 6 Data access time performance of pull clients and priority clients against the pull client's request arrival rate .

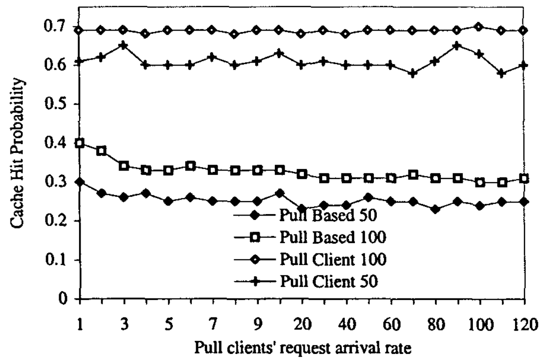

Fig. 7 Cache hit probability vs pull client's request arrival rate.

References:

[1] Chi-Jiun Su and Leandors Tassiulas, Broadcast Scheduling for Information Distribution, in Proceedings of Infocom'97.

[2] Veena Gondhalekar, Ravi Jain and John Werth, Scheduling on airdisks: Efficient access to personalized information services via periodic wireless data broadcast, in Proceedings of ICC'97.

[3] Demdt Aksoy and Micheal Franklin, Scheduling for Large-Scale On-Demand Data Broadcasting, in Proceedings of Infocom'98.

[4] Wang-Chien Lee and Dik Lun Lee, Signature Caching Techniques for Information Filtering in Mobile Environments, Wireless Networks, May, 1999.

[5] Jin Jing et. al. Bit-Sequences: An Adeptive Cache Invalidation Method in Mobile Client/Server Environments, Mobile Networks and Application, February, 1997.

[6] T. Imielinski, S. Viswanathan and B.R. Badrinath, Data on Air: Organization and Access. IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 3, 1997.

[7] Chi-Jiun Su and Leandors Tassiulas, Broadcast Scheduling for Information Distribution, Wirless Networks, May, 1999.

[8] Swarup Acharya, Micheal Franklin and Stanley Zdonik, Balancing Push and Pull for Data Broadcast, in Proceedings of ACM SIGDOM, May, 1997.

[9] Kian-Lee Tan and Jffrey X. Yu, Energy Efficient Filtering of Nonuniform Broadcast,

[10] Michael Franklin and Stan Zdonik, "Data In Your Face": Push Technology in Perspective, in Proceedings of ACM SIGDOM, June, 1998.

[11] Michael Franklin and Stanley Zdonik, A Framework for Scalable Dissemination-Based Systems, in Proceedings of ACM OOPSCA October, 1997.

[12] Azer Bestavros, AIDA-Based Rael-Time Fault-Tolerant Broadcast Disks, in Proceedings of Real-Time Technology and Applications, 1996.

[13] Yon Dohn Chung and Myoung Ho Kim, Energy Indexing for Wireless Broadcast Data, technique report of Department of Computer Science of KAIST.

[14] http://www.cs.umd.edu/projects/bdisk/papers.html