Broadcast on Demand:
Efficient and Timely Dissemination of Data in Mobile Environments

Ping Xuan, Subhabrata Sen, Oscar González, Jesus Fernandez and Krithi Ramamritham
Department of Computer Science, University of Massachusetts
Amherst, MA 01003
{pxuan, sen, ogonzale, jefer, krithi}@cs.umass.edu

Abstract

The demand for efficient, scalable and cost effective mobile information access systems is rapidly growing. Radio frequency broadcast plays a major role in mobile computing, and there is a need to provide service models for broadcasting information according to mobile users' needs. In this paper we present a model called Broadcast on Demand (BoD), which provides timely broadcasts according to requests from users. Compared to static broadcast, this approach has a different emphasis: it is based on a demand driven framework, aimed at satisfying the temporal constraints of the requests, and uses scheduling techniques at the server side to utilize the limited bandwidth dynamically and efficiently. In this paper, several broadcast transmission scheduling policies for BoD are examined. The study indicates that EDF-based policies combined with batching of requests achieve good performance. The results show that BoD is successful in satisfying the temporal constraints of the requests and is a viable service model for wireless broadcast stations.

1 Introduction

In the evolving field of mobile computing, there is a growing concern to provide mobile users with timely access to large amounts of information [5]. Examples of such services include weather, highway conditions, traffic directions, news and stock quotes. Due to the intrinsic constraints of mobility such as bandwidth restrictions, power consumption, connection reliability, the design of an efficient, scalable and cost effective mobile information access system is a challenging task.

The fast growing wireless communication industry has made digital wireless communication a reality. It is possible that in the near future, radio frequency electro-magnetic waves will be the most available communication medium for mobile users, and more often than not, the *only* accessible medium. This also means the rapid development and deployment of digital radio-frequency broadcast stations. Ideally, the services to be provided by these broadcast stations should allow the mobile users to access any information, anytime and anywhere; in reality, the broadcast stations will have their own service goals. Some of them may be dedicated for special purposes, some providing commercial subscription-based services to the public, some open to the public freely, and others providing a combination of services.

In addition to the different types of services, digital wireless communications allows a station to depart from the conventional *static broadcast* model to a more flexible scheme, which we call *dynamic broadcast*. By "static broadcast" we mean a broadcast where the schedule of programs is fixed, and even though the contents of a program can change with time, direct feedback from the users is not supported. An example of a static broadcast is a traditional weather information service. In such a service, the contents of program change between broadcasts in order to reflect the latest climate conditions. In contrast, in "dynamic broadcast", both the schedule of programs and its contents can change and there exists limited support to handle user's requests. Under this scheme both push and pull of information is possible. The design and evaluation of a service model suitable to support different types of services under a dynamic broadcast scheme forms the goal of this paper.

We propose a service model called *Broadcast on Demand (BoD)* that makes efficient use of the available bandwidth and allows users timely access to information. This model allows mobile clients limited ability to transmit queries to the server, as well as specify the maximum tolerable latency (i.e., deadline) allowed for a particular request. As a result, the information server is capable of producing a broadcast that not only satisfies the request of its clients, but at the same time is capable of retaining the benefits (high bandwidth utilization, high scalability, etc.) associated with broadcast communication. In this paper, we develop and evaluate several demand-based broadcast transmission scheduling policies.

The underlying idea behind the BoD model is to combine broadcast communication with on-demand data dissemination. The broadcast component of the BoD model provides flexibility in the scheduling of the response to a particular request and takes advantage of the benefits of static broadcast. Its interactive/on-demand component allows a mobile client to access any data item under temporal constraints. In effect, BoD tries to customize the broadcast to service individual clients better, but avoids the scalability problems of a client/server model. This is accomplished by intelligent scheduling at the server and using the broadcast nature of the medium, one broadcast satisfying many requests for the same data, leading to an efficient utilization of the bandwidth.

Our model is in contrast to the model of *Broadcast Disks* [1], and in some ways our model can be viewed as encompassing Broadcast Disks, since it exploits the characteristics inherent in broadcast communication. A Broadcast Disk involves periodic and continuous broadcast of data by a stationary server. It is suited for disseminating information to a large number of clients in a mobile environment, where energy and bandwidth efficient solutions are needed.

Under the conventional broadcast disk model, the server takes advantage of the available bandwidth to broadcast as much information as possible to clients, but it needs to decide what information to broadcast and at what rate, in order to compensate for the lack of communication from its clients. The lack of direct interaction between the server and the mobile clients gives rise to several fundamental problems, given the broadcast disk model.

First, for a mobile client it is impossible to access any data item which has not been selected for broadcast by the server. Secondly, once a broadcast schedule is decided, the server is "blind" — it broadcasts selected information using the whole bandwidth even if there is no user listening to it. Third, access to information with temporal validity properties is not guaranteed. Finally, the conventional broadcast disk tries to plan the broadcast off-line according to statistics gathered about user access patterns. While this may be good for accesses requiring the most popular data, it is likely to perform poorly for less frequently requested information. Such requests will typically experience long waiting times, which could violate the temporal constraints associated with requests.

The timing constraints that users specify in their requests arise due to various reasons. First, the information may be constantly changing, or may have a reduced value if not used in a timely fashion. Examples include information that affects decision making, such as current traffic conditions, airport informations services, stock quotes, etc. Also, since the latency is an important quality of service measure, the timing constraints can represent the maximum tolerable latency for a client. In this paper, we consider requests with such timing constraints, and we use the term *deadline* to refer to this constraint. These motivate us to propose the BoD model, which emphasizes the on-demand nature of broadcasts.

The remainder of this paper is organized as follows: In section 2, we detail the characteristics of the BoD model and relate it to work on broadcast disks. In section 3, we describe various approaches for scheduling data dissemination from the server. The simulation framework used for evaluating the different broadcast server scheduling policies is described in section 4. Section 5 presents the results obtained and their analysis. We conclude by outlining future work in Section 6.

2 The BoD model and Related Work

For traditional information dissemination, two models have been adopted previously:

- *Client/Server* — this serves a single client at a time (unicast), in an on-demand fashion. The advantage of this model is that it can provide timely, customized service, but it can suffer from bandwidth constraints severely. An on-demand client/server model is likely to perform poorly as it does not take advantage of the aggregation or batching possibilities inherent in a broadcast environment. Per request transmission will result in a unicast usage of the broadcast medium, which is bound to fail under overload conditions.

- *Broadcast Disk* — this serves multiple clients at a time (broadcast), in an autonomous and periodic fashion. This model provides efficient, low-power access to a large amount of information, but it lacks interaction, and does not deal with temporal constraints. Given the limited bandwidth and the *periodic transmission*, some data may be broadcast on rare occasions. Thus it is generally not suited for data accesses that have tight timing constraints, such as in real-time communication and computing.

In general, the problem of data transmission can be viewed along two dimensions (Figure 1). The first dimension ranges from *unicast* (broadcast for one client) to *broadcast* (broadcast for all possible clients). The second dimension differentiates between *static* (with periodic schedule) and *on-demand* broadcasts. Client/Server interaction can be viewed as the combination of on-demand and unicast, while broadcast disk is the combination of static and broadcast.

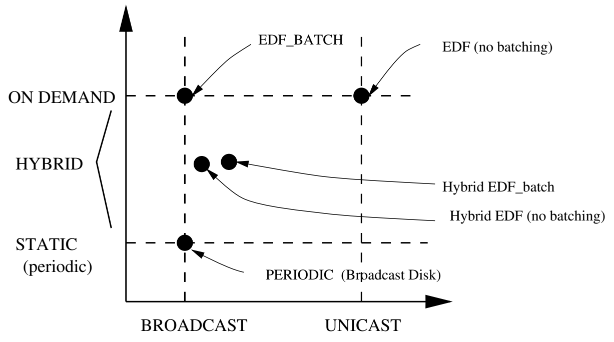

Figure 1. Candidate Transmission Policies

The figure also shows that other combinations are possible:

- *static/unicast*: Server provides periodic data to a single client at a time. In this model, data is sent even if the client does not need it. This does not seem to be a useful combination.

- *on-demand/broadcast*: Server provides on-demand data to multiple clients to satisfy a particular request. This is worth exploring and is part of the domain covered by BoD.

Our current approach to BoD also includes a hybrid combination, which sits between *static* and *on-demand* broadcast. Time division multiplexing is used at the server, in order to utilize a fraction of the transmission bandwidth for periodic broadcast, as in *broadcast disks*. The remaining bandwidth is used for on-demand processing. To exploit the broadcast potential of the medium, *batching*, which refers to the technique whereby the server responds to multiple requests for the same information by broadcasting the information only once, may be combined with *on-demand processing*. In this context, BoD includes broadcast disks as a special case of the time division wherein all of the bandwidth is allocated for periodic broadcasts. In this paper, we examine the mechanisms used to support the BoD model by evaluating five scheduling policies, shown as black dots in Figure 1.

By combining the different approaches, BoD attempts to get the best of all worlds. Some of the potential benefits of the BoD model include:

- *Satisfaction of temporal requirements* — by using deadline cognizant scheduling approaches (e.g. Earliest Deadline First (EDF)).

- *Effective use of bandwidth* — by only transmitting data that are requested, and by exploiting batching possibilities.

- *Customizing broadcast* — by choosing the scheduling policy that is most suited for a particular access pattern.

In the BoD model, we assume separate uplink and downlink channels. Mobile clients transmit short requests to the server for data along the uplink. For simplicity, data is requested using unique identifiers, such as a internet URL address, stock ticker number, etc. In addition to the identifier, a request is associated with a *deadline*, which refers to the maximum tolerable latency for that request. The deadline could be user-specified, or based on the QoS (Quality of Service) specification of the service subscribed to by user at the server. We also assume high data availability at the server, and assume that scheduling overheads at the server are negligible. The content of data may change constantly, and the server always tries to send the latest version available. In this paper we don't address the problem that data may "expire" at the server. In such case, we assume that the server is provided new data by sensors, etc.

We are primarily concerned with the service provided through the downlink. In response to the requests received from the uplink, the server uses some policy to schedule data transmission to satisfy the requests. Unlike pure client-server interactions, BoD uses scheduling policies which attempt to efficiently utilize the limited bandwidth by exploiting the batching possibilities offered by a broadcast environment. We consider simple scheduling policies for the periodic and on-demand broadcasts, and evaluate the performance of the BoD model. Our goal is to maximize performance with respect to satisfaction of deadline constraints and to achieve efficient use of available bandwidth.

Obviously, there are some tradeoffs involved in achieving the above benefits. For example, since clients have to transmit requests, the bandwidth needed for uplink facilities increases, but the latency of the data decreases. An adequate balance between these tradeoffs can be achieved partly by choosing the appropriate scheduling policy at the server side (the next section describes possible scheduling policies), and partly by adopting techniques suggested in the context of other research work related to wireless broadcast. Based on the specific application scenario, the techniques used for improving performance in traditional broadcast can also be deployed under the BoD model.

For example, when the broadcast schedule is known (or even partially known), indexing can be used to provide power-efficient access. Indexing of data within a broadcast disk has been addressed previously in [5, 6, 7]. Organization techniques that multiplex index information together with data on the wireless communication channels allow a mobile client to minimize the energy spent while tuning and keeping the access time under control. By incorporating the appropriate indexing techniques into the BoD model, we can address the power consumption restrictions of mobile users.

When the requests have locality, broadcast information can be organized in a way that improves access time for "hot" data, exploiting the information extracted from the monitoring of user access patterns to dynamically adjust the data to be broadcast. This technique, temporal replication [4], broadcasts more popular data items with a higher frequency than "cold" ones, so overall average data access latency is minimized (accesses to hot data experience shorter delays).

When the data at server side is not being updated often, caching and prefetching can be applied to achieve better data availability. In [8], prefetching and client cache management issues are addressed. BoD can benefit from caching in such scenarios as well, since more availability inherently means less requests and faster response times. In this paper we consider highly dynamic data such as stock data, traffic information, so caching may not be very effective in such scenarios.

In order to adapt to the ever-changing needs of the client population, most techniques rely on the monitoring of user request patterns or obtaining user access profiles from time to time. Without direct requests from the clients, it is very difficult to quickly adjust the broadcast to the client needs. As a result, in [2], broadcast disk architecture has also been extended to allow explicit requests from clients, thus moving from push-only service to a balance of push and pull.

More recently, broadcast disk organizations that try to satisfy deadline constraints have been proposed in [3]. This work discusses techniques to allocate data items to broadcast disks so as to mask the impact of intermittent failures, and avoid the long delay for waiting for server retransmission. Specifically, the paper considers three possible organizations for broadcasts: flat, rate-monotonic and slotted rate monotonic. These fault-tolerance techniques make broadcasts more reliable and predictive, and thus more suitable for real-time applications. This technique can also be applied under BoD to provide more reliable transmission. Combined with the ability to re-send requests at the client side, BoD can offer more flexible, and less costly fault-tolerant services to the clients.

3 Scheduling of Transmissions in the Broadcast on Demand Model

Choosing a good scheduling policy for the server is of central importance in BoD. Assume a request has an associated deadline in addition to the ID of the requested data item. The deadline associated with the request represents the maximum delay the mobile client is willing to incur while waiting for a particular data item. The *hotness* of a data item represents how frequently it is requested.

We consider 5 different *candidate* policies for broadcast scheduling, they are *PERIODIC*, *EDF*, *EDF_BATCH*, *Hybrid EDF*, and *Hybrid EDF_BATCH*, as depicted in Figure 1.

3.1. Periodic Broadcast

In periodic scheduling model, we examine how the static broadcast performs when clients need certain data and have deadlines associated with the data. Here the server is cognizant of the hotness of the data item through some initialization process, but has no interaction with clients otherwise, which means it is not directly listening to each request. Thus the deadlines associated with individual requests are not known, as in a typical broadcast disk. In the simulation, each request is logged so deadlines misses can be counted.

Suppose we have N data items on the server, each needing m packets to transmit. Associated with a traditional broadcast, there is a notion of *broadcast length*, or the period of the broadcast. At the end of each period, the server repeats its broadcast schedule. Also, suppose within the broadcast length, data items are broadcast $n_1, n_2, \ldots, n_N$, times, which reflect the distribution of *hot* (more frequently requested) and *cold* (less in demand) data. Then, the total number of data items (counting copies) is $L = \sum_{i=1}^{N} n_i$, and the broadcast length is $L \times m$.

The scheduling problem at the server is: given N data items, schedule their broadcast so as to maximize the number of clients requests satisfied before their deadlines. Since requests may arrive at any time, we should try to place all n_i copies of the *i*'th data item *evenly* separated within the layout of the total L data items. In theory, such evenness means that for the i'th data item, if the n_i copies are placed at $a_1, a_2, \ldots, a_{n_i}$, we need to minimize $\Delta_i = \sum_{j=2}^{n_i}(a_j - a_{j-1} - L/n_i)^2$, since L/n_i is the ideal separation for consecutive broadcasts of the same data item. To achieve such optimality is computational intensive, so we use the following simple yet effective algorithm in our experiments:

First, we choose a number B, where $B \geq L$, preferably a common multiple of all n_i, so that B/n_i will be an integer, thus the error caused by rounding to integer can be reduced.

The idea is that first, we try to layout all copies of all data items evenly within B; of course since $B \geq L$, there will be empty slots, so the next step is to compact the layout by eliminating empty slots, and thus obtaining a tighter layout of size L. We do not use L when the first layout is created because it is easier and quicker to find an empty slot for an item if a larger size template (with B slots) is used, thus data items will be placed more evenly.

The steps to create such a layout are:

1. Begin with an empty list of size B, which is the template for the layout. Let the current position c be the first slot in this list.

2. Select i, the next data item to be laid out, (starting from data item 1).
3. Lay out i, until all n_i copies of i have been laid out, according to the following: if the current position c is occupied, assign c to the next empty slot. If the end of list is reached, wrap around to the beginning of the list and assign the next empty slot to c; then layout i at c; and finally increase c by B/n_i. (The reason is that, ideally, for data item i, two adjacent copies of i should be separated by B/n_i in the un-compacted layout. However, in the event that the ideal place is occupied by other data items previously laid-out, we try to search the next place. Since B is large, (at least $\geq L$), statistically, the next empty place can be found quickly.)
4. Repeat steps 2 and 3 until all N items have been laid out.
5. Compact the list towards the beginning of the list by eliminating empty slots between data items.

Similar, but more complex layout alternatives, such as the layout discussed in [1], could also be used to meet the same problem specification. In the future we are planning to examine if they can lead to any performance gains.

3.2. EDF Broadcast

For data items requested by mobile clients which have to be served via on-demand broadcasts, the server has to schedule their transmission within a certain deadline associated with the request. As a baseline *on-demand transmission scheduling* policy, we consider planning-based non-preemptive EDF as described in Figure 2. The scheduler schedules the requests with earliest deadline first, and rejects those requests that are not schedulable (i.e., have too tight deadlines.)

This policy treats requests for different data items more fairly than the purely periodic policy. It tries to schedule transmissions based purely on the deadline of the request and is completely oblivious to the relative popularity of the requested data. As such, requests for comparatively colder items will have a greater probability of being satisfied as compared to a purely periodic broadcast based only on access frequencies. That is, under EDF, these items appear much more accessible to the clients. The server only transmits data to satisfy some request, so it never wastes any bandwidth, unlike the periodic policy, under which some transmitted item may not be needed by any client.

3.3. EDF-Batch Broadcast

The *EDF* broadcast policy schedules a separate transmission for each request it can handle. In particular, hot items are transmitted once per request, so it is expected to perform poorly under overloaded situations which can occur often in bandwidth-constrained situations such as mobile environments. It is easy to see that there is no need to send the same information multiple times if the first transmission can satisfy subsequent requests.

Procedure LayoutEDF (t)
$Target$ = set of requests which need to be scheduled.
t = earliest time slot at which scheduling can occur.
  $Scheduled$ = 0;
  / $Scheduled$ is the list of requests in $Target$ which have been scheduled */*
  Sort $Target$ in order of increasing deadlines.
  While($Target$ is nonempty)
    Choose the request i with the closest deadline.
    D_i = the data item requested by i.
    size(D_i) = the size of the data requested by i.
    T_i = the deadline associated with i.
    if (t+size(D_i)$\leq T_i$)   */* schedulable */*
      reserve interval (t, t+size(D_i)) to broadcast D_i;
      t += size(D_i);
      add i to $Scheduled$;
    else reject request i;
END Procedure

Figure 2. Algorithm for Planning-based EDF

In particular, the use of *batching* multiple requests for the same data item at the server side, and handling the batched requests via a single transmission of this data item allows more efficient utilization of the available bandwidth and can potentially increase the effective server capacity by allowing more requests to be served. This also handles overloads very efficiently. This yields our next policy called *EDF_BATCH*.

Just like the EDF policy, EDF_BATCH also tries to lay out a transmission schedule for the different requests based on *earliest deadline first*. However, unlike EDF, for each candidate request, it checks to see if a transmission has already been planned for the requested data. If so, it does not re-send the same data, as the planned transmission will be able to satisfy the current request also. Only if no transmissions have been planned for the requested data, does the scheduler attempt to plan a new transmission for this request. The bandwidth thereby saved can be used to satisfy other requests. The additional overhead over EDF is very small, involving the maintenance and checking of a flag for each data item. Also note that this approach, while attempting to serve a group of requests via a single transmission does not introduce any additional delays for the clients involved. As such, it is expected to perform at least as well as non-preemptive EDF in satisfying client requests.

3.4. Hybrid Broadcast Policies

As the name suggests, the *hybrid broadcast policies* incorporate both periodic as well as on-demand broadcast strategies. The data which is expected to be accessed most often is periodically broadcast using a fraction of the available channel bandwidth. The periodic broadcast is laid out using the *Periodic policy* outlined before. This allows different broadcast frequencies for hot/cold data. The remaining part of the channel bandwidth is used for servicing requests which cannot be satisfied by the periodic broadcast.

The method used for multiplexing periodic and on-demand broadcasts from a single server is based on time-division broadcasting. The broadcast server assigns a fraction of the total available bandwidth for the periodic broadcast, and the rest of the bandwidth is assigned for on-demand broadcast. We can think of the broadcast server as two separate logical servers, the first dedicated to broadcasting the periodic data items and the second serving the requests not satisfied by the periodic transmission.

When a client request arrives, the server checks if it can be satisfied by the periodic broadcast. If the broadcast server determines that the requested data will be transmitted by the periodic broadcast within the deadline associated with this request, no explicit scheduling action is needed. Otherwise the request needs to be handled using the bandwidth allocated for on-demand broadcasts. Therefore, the on-demand broadcast is used to satisfy requests for

- relatively cold data which are not expected to be accessed frequently and therefore are not broadcast periodically.
- data which is part of the periodic broadcast, but, the next scheduled transmission of the data will not satisfy the deadline associated with the request.

For the on-demand part of the broadcast channel, different policies are possible:

- *Hybrid EDF Broadcast* uses EDF for the on-demand broadcast.
- *Hybrid EDF_BATCH Broadcast* uses EDF_BATCH for the on-demand broadcast.

4 Simulation of the BoD environment

In order to understand and evaluate the properties of the broadcast on-demand model, we have implemented a simulation model of a BoD environment. In this section we describe the simulator, the performance metrics, and assumptions made in generating both the data set and the load in our experiments.

The primary performance metric of interest is:

SR (Success Ratio) : The number of requests successfully served by the BoD server as a percentage of the total number of incoming requests.

For the hybrid policies, we also measure:

PRP : Percentage of requests successfully serviced by the periodic broadcast.

PRO : Percentage of requests successfully serviced by on-demand broadcast.

Note that **SR = PRP + PRO**.

4.1. Data and Request generation

A parameterized workload generator creates the data set and the entire trace workload before each simulation run. The workload corresponds to the requests from mobile users. We assume that the bandwidth of the broadcast channel is 1Mb/s. Some important simulation parameters are:

- **Exponential distribution of interarrival time**: the interarrival time of client requests is assumed to have an exponential distribution with mean *x*.
- **Packet size**: the size of a packet in bits. We consider small packet sizes, equal to 500 bits (1/2000 of bandwidth). The unit of time in these experiments is the time to transmit one such packet (0.5 ms).
- **Number of data items**: the number of distinct data items available to the clients. Set to 1000.
- **Size_data_item**: the size of each data item in packets. Set to 10 packets.
- **Perc_per_channel**: percentage of the total bandwidth allocated for periodic broadcast.
- **Perc_aper_channel**: percentage of the total bandwidth allocated for aperiodic broadcast (1 - Perc_per_channel).
- **Max_laxity**: the maximum laxity of any request. We run separate simulations for Max_laxity = 500, 1000, 2000 and 6000 time units (250ms, 500ms, 1s, and 3s respectively).
- **Min_laxity**: the minimum laxity of any request. Set to 200 time units (100 ms). This is comparable to the broadcast period of the *hottest* data item, so that the server is not broadcasting too often (i.e., not broadcasting with a period that is much shorter than deadlines associated with requests).

The interarrival time of requests is used to control the load for the BoD server. The deadline of a request is given by (Current_time + time to transmit the data item (i.e., 10 time units) + V), where V is a random number selected uniformly from the range (Min_laxity, Max_laxity). Since the time to transmit the data item is very small compared to Min_Laxity, in the following text we loosely use *deadline* for *laxity* when we do not need to emphasize the difference between laxity and relative deadline.

We assume that the server is cognizant of the relative *hotness* or *coldness* of the different data items. This could be computed by the broadcast server based on past access patterns. In our simulations, we assume that the different data items have associated access probabilities governed by Zipf's law [9]. The data item requested by a client is modeled using the same Zipf distribution, i.e., we model the case where the actual access patterns match the expected access behavior. Zipf's distribution says that when N items are available, the probability of choosing item i is proportional to $1/i$, thus the first few items are much more likely to be chosen than the last few items — therefore data items are associated with "hotness" in this distribution.

4.2. Simulator

The simulator models a single broadcast server. It is time-driven, with the time measured in broadcast units (the time required to transmit a single data unit).

For the purely periodic and hybrid policies, the layout of the periodic broadcast is determined using the *Periodic policy* discussed before, and mapped onto the time slots reserved for periodic broadcast.

The scheduling algorithm runs every 100 time units. If in between 2 invocations, the broadcast channel falls idle, the scheduler is run again. The scheduling overhead is not accounted for in the simulation.

In all the experiments, the simulation is performed over one million time units (500 s). Each point in the graphs is the average over 6 simulation runs. The variances for these averages were computed and they were less than 2% relative to the values of averages.

5 Experimental Results

As described previously, we ran simulations for all 5 scheduling approaches. We discuss the results in the same sequence as they appear in section 3. In all the figures, X-axis represents the average interarrival time. Given the parameters in the previous section, each data item needs 10 time units to transmit, so an average interarrival time x corresponds to a total workload of $(10/x) \times 100\%$ the channel capacity, so the figures show results for workloads ranging from 1000% channel capacity (x=1) to 100% channel capacity (x=10). The Y-axis depicts the success ratio (SR), or the percentage of requests successfully serviced by periodic/on-demand broadcast subchannel, when hybrid broadcast is used.

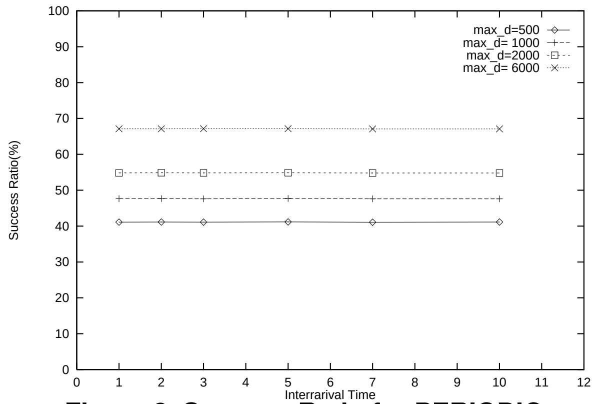

Figure 3. Success Ratio for PERIODIC

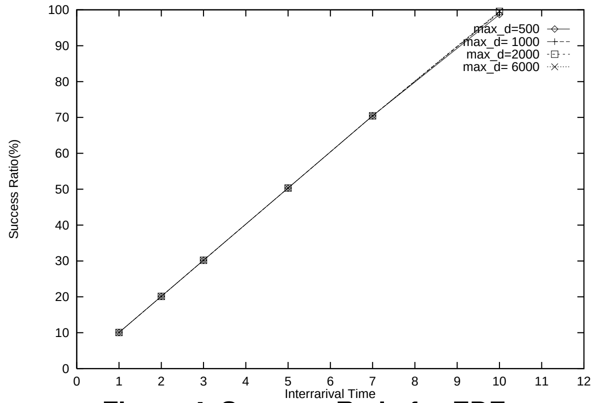

Figure 4. Success Ratio for EDF

5.1. Pure periodic broadcast (PERIODIC)

Figure 3 demonstrates how traditional broadcast performs when requests have deadlines, using the scheduling policy described in section 3.1.

As the deadline distribution (in our simulation, a uniform distribution between minimum and maximum deadline) changes, SR varies significantly. The longer the maximum deadline (max_d), the higher the SR.

Due to the fact that pure periodic broadcast is highly scalable — one broadcast can serve an unlimited number of simultaneous requests, SRs do not depend on workload, hence the curves are parallel to x-axis. This is the advantage inherited from uni-directional broadcast.

5.2. Pure On-Demand broadcast, without batching (EDF)

Here in Figure 4 we show how this baseline broadcast model handles the requests. Using EDF-like scheduling, deadlines are taken care of, but the distributions, or characteristics of the requests are ignored by this scheduling model. Thus, in overload conditions, such a simple client/server

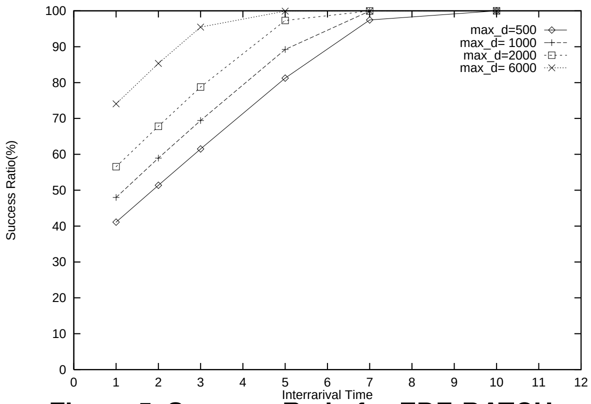

Figure 5. Success Ratio for EDF_BATCH

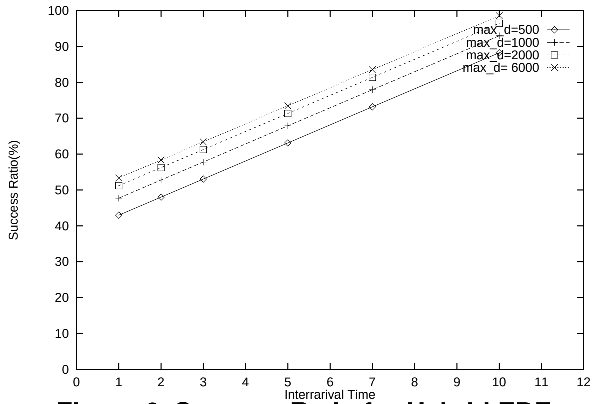

Figure 6. Success Ratio for Hybrid EDF

model cannot scale beyond the capacity of the broadcast channel, i.e., SR $\leq$ (capacity/workload) = $x \times 10\%$.

On the other hand, since deadlines of requests are uniformly distributed between 200 and some maximum value (max_d in the figures), the deadline laxities are fairly large. Therefore, the scheduler is capable of keeping the broadcast channel busy all the time in overload conditions. Thus, the broadcast bandwidth will always be utilized, and SR $\approx x \times 10\%$ is observed. This is why the data points for different max_d overlap with each other. In this case, success ratios are not sensitive to the maximum deadline.

When $x = 10$, the SR is almost 100%. Intuitively, this indicates the advantage of using on-demand model: it can provide fast response when the workload is light.

5.3. Pure On-Demand broadcast, with batching (EDF_BATCH)

When batching is added to the previous case, we see significant performance gains (Figure 5). When $x = 1$, the success ratio increases to 41.5%, compared to only 10% in the previous case. This success ratio is also slightly better than in the pure periodic broadcast case, where the SR is 41%. When the maximum deadline increases from 500 to 1000, and from 1000 to 2000, the SR also increases, roughly 7.5% when the maximum deadline doubles – this is far larger than the 3% increase in the pure periodic broadcast case. When x increases , the SR increases as well, reaching 100% at $x = 10$. This is because, due to batching, the effective load is much lower than the nominal workload obtained by summing up all the requests. Since requests have quite lax deadlines, many requests can be batched together and served by just one broadcast. The longer the maximum deadline and the shorter interarrival time, the more effective batching is. In fact, when maximum deadline = 6000, SR is very close to 100% even when the nominal workload is 200% (interarrival time = 5).

The effectiveness of batching depends on the distribu-

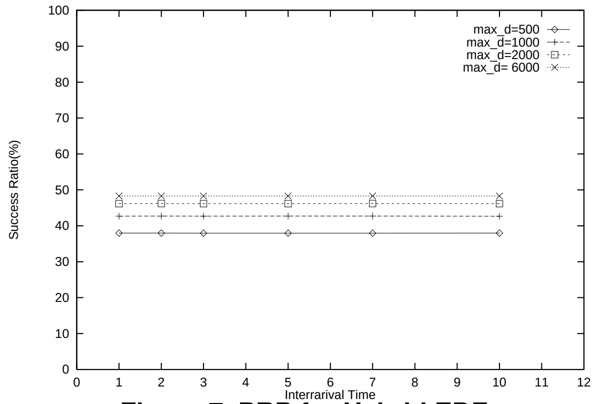

Figure 7. PRP for Hybrid EDF

tion of requests. Specifically, there are two aspects that may affect the performance: how *often* and how *bursty* is the request for a data item? As explained in the previous section, we use Zipf distribution as the distribution of the data item being requested — this results in a heavily skewed distribution towards the *hot* end. The request interarrival time generated according to exponential distribution, which is independent of the selection of data item in the previous request — results in no inherent burstiness. Since batching is associated with the temporal locality of request, the more often and the more bursty the requests are, the more effective batching can be.

5.4. Hybrid broadcast, without batching (Hybrid EDF)

In case of the hybrid policies, we assume that 50% of broadcast channel is used for the periodic broadcast and the remaining 50% is used for on-demand broadcast. An important question then is, what should go into the periodic broadcast. We consider a simple *proportional reservation policy* whereby we choose for inclusion, the m hottest data items which together account for 50% of the expected access requests, as indicated by the corresponding Zipf dis-

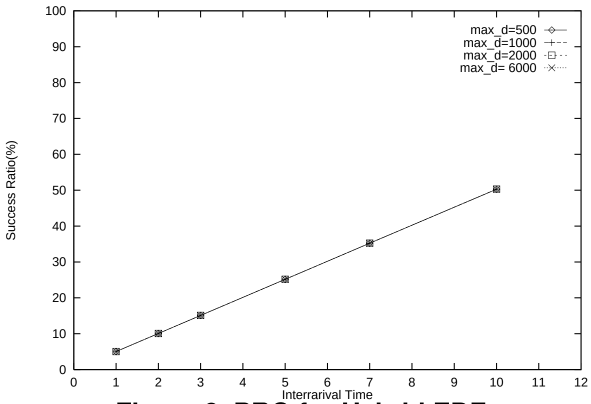

Figure 8. PRO for Hybrid EDF

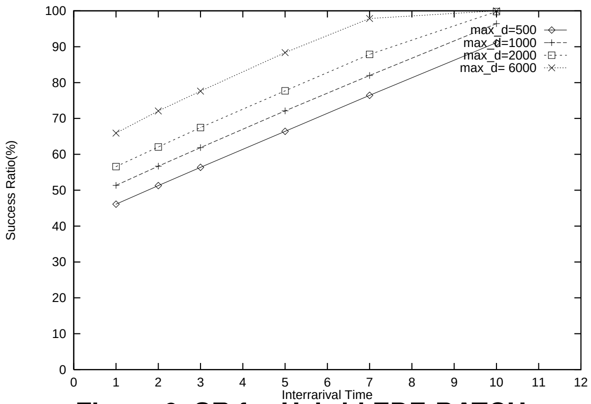

Figure 9. SR for Hybrid EDF_BATCH

tribution. The periodic and on-demand broadcasts are multiplexed onto the broadcast channel by reserving alternate time units for each type of transmission.

Figure 6 shows the total success ratio (SR), which is the sum of the PRP (Figure 7), and the PRO (Figure 8).

According to Figure 6, we see that SR increases roughly linearly with respect to interarrival time x. We notice that, in general, the success ratios are much better than both PERIODIC (Figure 3) and EDF (without batching – Figure 4). *This indicates the advantage of using the hybrid approach.*

However we can also see that in some places, the total success ratio is smaller: for example, at maximum deadline = 500, the SR is 43% at $x = 1$, which is better than both PERIODIC (41%) and EDF broadcast. However, when maximum deadline increases to 6000, with same $x = 1$, the total SR is 53%, smaller than the SR for PERIODIC. Also, when interarrival time is 10 — in EDF, SR is very close to 100%, but here SR is a little smaller (ranging from 88% to 99%), and varies with the values of maximum deadlines. These can be explained by studying Figures 7 and 8.

In Figure 7, we again see the curves to be horizontal, a characteristic of periodic broadcasts (uni-directional). However, although only 50% of the bandwidth is used here, we note that when the maximum deadline is 500, PRP is 38%, very close to the SR we have for PERIODIC (41%), where 100% of the bandwidth is used. Also, since we broadcast only the *hot* data items – those data items that account for 50% of all the requests (in our case, the hottest 22 data items out of 1000 possible data items), this tells us that periodic broadcast is very effective for *hot* items, and possibly, *only* effective for *hot* items. The bulk of the SR is achieved by the broadcast of the hottest data items. The PRP increases with maximum deadline, but this increase is rather slow, since 50% is the limit.

In Figure 8, we again see the characteristic of pure on-demand broadcast – the curves are not sensitive to maximum deadline, so the four curves overlap with each other. The explanation to this effect is that, the 50% bandwidth for on-demand broadcast is fully utilized under overload condition (even when $x = 10$, since whatever cannot be handled by periodic broadcast is piped into on-demand broadcast), so PRO appears to be exactly half the SR in EDF (Figure 4).

Therefore, we see that although using the hybrid approach gets the best of both PERIODIC and EDF in most cases, when the workload is very heavy, the inability to handle *cold* data items in PERIODIC causes the increase of total SR when maximum deadlines are increased to be small. At the same time, the inability to handle overload in EDF causes the total SR to be below 100% when interarrival time is 10.

5.5. Hybrid broadcast, with batching (Hybrid EDF_BATCH)

Figure 9 shows the total success ratio (SR). The PRP is identical to that for Hybrid EDF (without batching) (Figure 7), — because adding batching does not affect the periodic broadcast subchannel at all. The PRO shows some improvement (Figure 10). The total success ratio is higher than that of Hybrid EDF, and performs much better than PERIODIC. Compared to EDF_BATCH, the hybrid approach performs better when the workload is heavy and when maximum deadline is short (46% when $x = 1$ and maximum deadline is 500, compared to 41% in EDF_BATCH). EDF_BATCH still does a better job when the workload is lighter, since it has 50% more of the bandwidth available for client's requests (as shown is Figure 5). Under conditions of light load (interarrival time = 10), Figure 9 shows that the success ratio values for hybrid EDF_BATCH are quite close to the values obtained by EDF_BATCH.

Figure 10 shows the performance gain in the on-demand subchannel by using batching, compared to Figure 8, where batching is not used. First, the curves are not overlapping each other anymore when maximum deadline varies. At maximum deadline = 500, we see that there is approximately

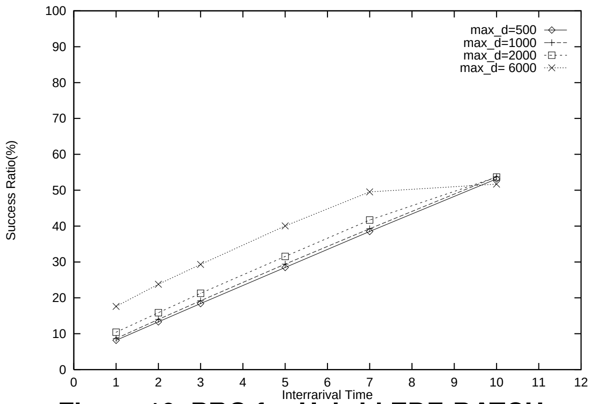

Figure 10. PRO for Hybrid EDF_BATCH

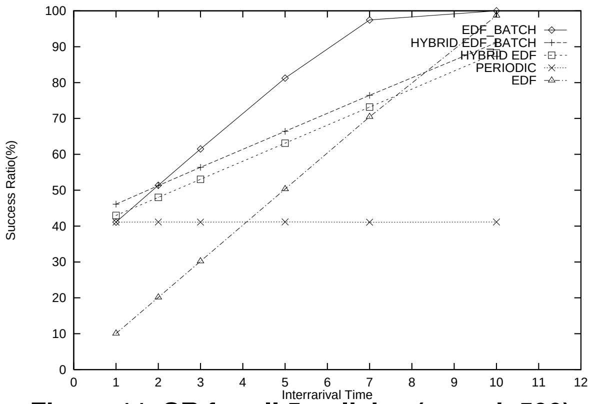

Figure 11. SR for all 5 policies (max_d=500)

3% gain compared to Figure 8, which shows the effect of batching for even small maximum deadline values. When maximum deadline increases, we see even better gain. At maximum deadline = 6000, we see about 7% gain compared to maximum deadline = 2000, which indicates that batching happens not only for frequently requested data items, but also for those data items that are not so *hot*.

There are two other observations worth making: first, we see that the percentage of requests handled by on-demand broadcasts can exceed 50%, as shown in Figure 10 when interarrival time is 10. This is because all requests that cannot be satisfied by periodic broadcast will be piped into on-demand broadcast, resulting in an overload for the on-demand subchannel. Since batching can handle overload caused by requesting the same data items, it is possible for the percentage successfully handled by on-demand broadcast to be higher than 50%. Indeed this is what is happening in our experiments.

Second, we see that the PRO at $x = 10$ and max_d=6000 is slightly lower than the cases when maximum deadlines are shorter. This is because when maximum deadline = 6000, the periodic broadcast satisfies 48.4% (Figure 7) of the total request, and thus there are less requests piped into on-demand broadcasts, where the on-demand broadcast with batching virtually handles them all, and achieves nearly 100% total success ratio (as seen in Figure 9).

5.6. Discussion of Simulation Results

To summarize the differences in performance of each of the scheduling policies, Figure 11 compares the success ratio for all 5 policies when maximum deadline is set to 500. This case represents short response time requirements.

Among the five candidate scheduling policies tested here, pure on-demand broadcast with batching and hybrid broadcast with batching are better than the other three alternatives. Pure on-demand broadcast with batching achieves the highest success ratio when the workload is not too heavy. When workload is very heavy (interarrival time is small), however, hybrid broadcast with batching performs the best, thus resulting in a crossover between EDF_BATCH and Hybrid EDF_BATCH. This is due to the high scalability of periodic broadcast — where the success ratio is insensitive to interarrival times (workloads). Since in most cases, the communication load is likely to be heavy, and the users will typically require fast response times — hence short deadlines, we believe that the Hybrid EDF_BATCH policy is well-suited for broadcast servers. For moderate to low overloads ($2 <$ interarrival time < 10), EDF_BATCH performs better than the hybrid policy, but the performance curves converge as the load decreases beyond a certain level ($a = 6$). The experiments do not account for the difference in scheduling overhead involved (as discussed in Section 6). It is conceivable that even for low to medium overload regions, the Hybrid EDF_BATCH policy (with its lower overhead) would be an attractive policy, for example, in low-end PC based broadcast servers.

6 Conclusion and Further Work

Our evaluations indicate that using relatively simple transmission scheduling policies, suitably adapted to make efficient use of the transmission bandwidth, BoD is a viable approach for satisfying client requests even under overload conditions. The flexibility of the BoD model allows the adaptation of the information being broadcast by the server to the requests being made by mobile users currently in its service area, and by choosing suitable scheduling policies, BoD can provide efficiency, adaption, and scalability.

We believe that the capability of dealing with deadline requirements in accessing specific data items is a needed feature for real-time mobile environments, and on-demand broadcast will be an important way to provide versatile, timely, and convenient services to mobile users. Future work will involve the development of more efficient and more scalable scheduling policies, and for optimal approaches to different application scenarios under the BoD model.

One issue that needs future work under this model is battery power. Research has shown the effectiveness of *indexing* in saving client power. Under the BoD model, index information can be transmitted from the server at some frequency, as an integrated part of the periodic broadcast, and the content of index may include broadcast times for either *hot* items, or a combination of both *hot* and *cold* data items. For example, in hybrid broadcast with batching, we could transmit indexes periodically, informing the clients of the broadcast frequencies of *hot* data items, and the broadcast times for *cold* data items. Also, when indexing is used, a client may not need to transmit a request if the needed data is scheduled for transmission within the deadline. Therefore transmission power can be effectively reduced by not sending such requests to the server, therefore reducing the chance that too many requests arrive simultaneously at server, improving scalability.

Also, the success of pure on-demand broadcast with batching indicates that if we reduce the bandwidth of the periodic broadcast subchannel — thus increasing the bandwidth available for the on-demand broadcast subchannel, we might be able to improve the performance of the hybrid algorithms. In our simulation, we adopted a simple *proportional reservation* policy — 50% of the server bandwidth was used to handle 50% of the total requests — the requests for *hot* data items. It will be instructive to study the performance if we allocate less, say 30% or 40% server bandwidth to handle the same 50% of the total requests, or more generally, to study how to optimally do the bandwidth division between the periodic and on-demand broadcasts. Again, the results will certainly vary according to the distribution of requests.

Since the distribution of requests may also vary over time, it will help if the division between periodic and on-demand subchannels is adaptive. In the current simulations, the interleaving of periodic and aperiodic broadcast is static. A dynamic time division broadcasting would allow a higher bandwidth utilization, as the fraction of bandwidth assigned to periodic/on-demand broadcast would be adjusted dynamically according to data access pattern — based on how *hotness* and *coldness* change over time, thus allowing the server to provide the best service at all times.

The current work does not address the difference in scheduling overheads involved. In particular, the EDF_BATCH policy (which has to consider all requests in EDF order) would have a higher overhead than the hybrid version (which has to apply EDF on only those requests not satisfied by periodic broadcast). While this is reasonable for high-end broadcast servers with fast processor(s), we need to address the scheduling overheads explicitly for less powerful systems.

In conclusion, *broadcast on demand* is a new, effective, and promising technique that expands the functionality of data broadcast, and provides a variety of digital wireless broadcast service models. Servers as well as mobile units can benefit from the versatile ability to handle both periodic and on-demand broadcast in a timely manner, and at the same time achieve efficient utilization of the limited bandwidth. Mobile clients can benefit from the real-time responses from the server and also the convenient ability of accessing any data via air. Although we still have ways to go in order to achieve the goal of providing timely data access anywhere, anytime, *broadcast on demand* is one sure step toward its realization.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, Broadcast Disk Data Management for Asymmetric Communication Environments, *Proceedings of ACM SIGMOD International Conference*, San Jose, California, May 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik, Balancing Push and Pull for Data Broadcast, *Proceedings of ACM SIGMOD International Conference*, Tucson, Arizona, May 1997.

[3] A. Bestavros, AIDA-based Real-Time Fault-Tolerant Broadcast Disks, *Proceedings of IEEE Real-Time Technology and Applications Symposium*, 1996.

[4] T. Chiueh, Scheduling for Broadcast-based File Systems, University of New York at Stony Brook, Computer Science Department Technical report, November 1994.

[5] T. Imielinski, S. Vishnatwan, B. R. Badrinath, Energy efficient indexing on air, *Proceedings of ACM SIGMOD conference*, Minneapolis, MN, May 1995.

[6] T. Imielinski, S. Vishnatwan, B. R. Badrinath, Power efficient filtering of data on the air, *Proceedings of EDBT*, LNCS, Springer-Verlag, 1994.

[7] T. Imielinski, Vishnatwan, B. R. Badrinath, Data on Air: Organization and Access, *IEEE Transactions of Data and Knowledge Engineering*, July 1996.

[8] S. Zdonik, M. Franklin, R. Alonso and S. Acharya, Are Disks in the Air Just Pie in the Sky?, *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December 1994.

[9] G. K. Zipf, Human Behaviour and the Principles of Least Effort, Addison-Wesley, Reading MA, 1949.