Adaptive Dissemination of Data in Time-Critical Asymmetric Communication Environments

Jesus Fernandez and Krithi Ramamritham
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
E-mail: jefer@cs.umass.edu, krithi@cs.umass.edu

Abstract

The proliferation of new data-intensive applications in asymmetric communication environments has led to an increasing interest in the development of push-based techniques, in which the information is broadcast to a large population of clients in order to achieve the most efficient use of the limited server and communication resources. It is important to note that quite often the data that is broadcast is time-critical in nature.

Most of the related current research focuses on a pure push-based approach (Broadcast Disks model), where the transmission of data is done without allowing explicit requests from the users. More recently, some bidirectional models incorporating a low-capacity uplink channel have been proposed in order to increase the functionality of the Broadcast Disks model. However, the impact of integration of the uplink channel has been investigated using only static client profiles or ignoring the existence of timing constraints associated with data. None of the existing models integrates all the characteristics needed to perform effectively in a real-world, dynamic time-critical asymmetric communication environment.

In this paper we present an adaptive data dissemination model and the associated on-line scheduling algorithms. These improve the functionality and performance of bidirectional broadcast models, maximizing the total number of satisfied users in asymmetric communication environments with dynamic client profiles and time requirements (e.g., mobile systems). This is achieved by means of dynamic adaptation of the broadcast program to the needs of the users, taking into account the bandwidth constraints inherent in asymmetric communication environments and the deadline requirements of the user requests. Performance is evaluated by simulation of a real-time asymmetric communication environment.

1 Introduction

There is a growing need for information servers with capabilities to handle large amounts of information accessed by a potentially unlimited number of users in asymmetric communication environments (e.g., traffic information, stock quotes, weather, news). The volume of information transmitted from the server to the users is much larger than the volume of information transmitted from the users to the server in such environments. In most cases, the users require the information by a certain time, and therefore, an adequate model should integrate the ability to serve information with deadline constraints associated with the requests of the data. We will refer to these as *time-critical asymmetric communication environments*. For example, let us consider a traffic information server and the driver of a vehicle who, at some point ahead in the road, needs to take one of two possible routes in order to get to her destination. Obviously, it is necessary for the server to provide the driver with the desired traffic information (for example, one of the routes is congested because there has been an accident) *before* the decision point is reached, otherwise the information has no value for the driver.

One way to try to meet the needs of such applications is to use the classic client/server model. Unfortunately, it suffers from a lack of scalability. Beyond a certain workload, the server becomes overwhelmed, that is, it is not able to serve more than a certain number of requests per time unit, due to resource and bandwidth limitations. In fact, this is becoming an important problem in the Internet world, where World Wide Web servers are based on a pull-based scheme and so cannot cope with the demands of an increasingly large population of clients. On the other side of the spectrum some bandwidth-efficient data broadcast models have been proposed [1]. They provide high scalability by broadcasting the information simultaneously to a large population of users, using a single, high-capacity downlink channel (server-to-users). The weaknesses found in pure broadcast models result from the absence of direct interaction with the users. Their two main problems could be stated as follows:

- When user's access patterns are not static (e.g., due to mobility of users), it is impossible to decide which data items have to be broadcast in order to satisfy as many users as possible at any given moment.
- In time-critical asymmetric communication environments, information must be received by a certain *deadline*. In general, data broadcast models provide no mechanisms to ensure that time requirements are met.

In an attempt to solve both problems, some bidirectional broadcast models have been studied [2, 17], trying to exploit on the one hand the benefits derived from the high bandwidth efficiency and scalability of data broadcasting, and on the other hand the direct interaction with the users of the client/server paradigm. Bidirectional schemes are based on the incorporation of a low-capacity uplink (users-to-server) channel, devoted to the transmission of requests made explicitly by those users not satisfied by the broadcast program. Most bidirectional broadcast models are *hybrid*, that is, there exist two different information transmission modes: the **periodic broadcast** mode, in which information is broadcast periodically, and the **on-demand broadcast** mode, devoted to broadcasting information explicitly requested by users via the uplink channel. Unfortunately, the only hybrid model in which deadline constraints are taken into account [17] is *not* adaptive, that is, the information that is broadcast periodically and the fraction of the bandwidth allocated to each mode does not change with time.

Consequently, none of the existing models is adequate for environments in which the user access distribution is not static and there are timing constraints present, which is the case in most real-world mobile systems. This paper addresses this lacuna. It introduces a time-critical *adaptive* hybrid broadcast model, that extends the functionality and performance of current hybrid models, by means of an on-line scheduling algorithm that takes into account the access frequency distribution of data, bandwidth limitations in both uplink and downlink channels, and deadline constraints associated with data requests. In order to minimize the overall *number of deadlines missed*, the information server dynamically adapts (a) the specific data items that have to be broadcast periodically, and (b) the amount of bandwidth assigned to each transmission mode. This adaptation of the *amount* and *contents* of the information being broadcast periodically to the data and time-criticality needs of the users at any instant in time dramatically improves performance over non-adaptive hybrid models in environments where dynamic access distributions are present.

The remainder of this paper is organized as follows: Section 2 states the main issues and considerations when designing an information server for a time-critical asymmetric communication environment. Section 3 discusses related work. The description of our adaptive hybrid data broadcast model is presented in section 4, in which the adaptive scheduling approach is also detailed. Section 5 presents and analyzes the results of the experimental study. Finally, section 6 summarizes and concludes the paper.

2 Issues and Considerations

Let us begin by explicitly stating the characteristics that an information server should include in order to perform effectively in time-critical asymmetric communication environments:

- *High bandwidth efficiency*: Due to the inherent bandwidth limitations in asymmetric communication environments, the server must use the available bandwidth with maximum efficiency, in order to satisfy as many users as possible (before their deadlines).
- *High scalability*: As the client population may grow without limits, we need a server model capable of satisfying an extremely large number of users simultaneously without incurring significant performance penalties.
- *Adapting to changing user profiles*: The client population will change with time in most situations (e.g., in mobile environments). This means that the server should be able to adapt itself to diverse access patterns.
- *Awareness of deadline requirements*: There are many applications in which the information must reach the user before a certain deadline in order to be useful. The server must incorporate appropriate mechanisms to take into account these timing constraints.

If we want to ensure that the first two characteristics are satisfied, a broadcast-based information server is the only reasonable option to consider. The key point is that by broadcasting information, the bandwidth is, in effect, multiply used, as a single transmission of a given piece of information can reach an unlimited number of users. In order to provide adaptiveness, the broadcast server should use the available bandwidth to satisfy as many clients as possible, by broadcasting the right information in each situation. At the same time, as user profiles may change, the server needs some feedback from the actual users to estimate the actual user access pattern, and consequently the broadcast server has to be bidirectional, incorporating an uplink channel – from users to the server. Thus, the server model must have the following capabilities (which, as it turns out, are satisfied by adopting a hybrid model):

- *Potential to use uplink bandwidth effectively.* Clearly, the items that are included in the periodic broadcast program do not have to be requested. Assuming that some index information is interleaved with the broadcast, users should consult the index to check if the information they need is already being broadcast periodically and if they would receive it on time, before sending a request to the server. This decreases the number of requests that have to be sent to the server, decreasing the probability of saturation of the uplink channel.
- *Potential to use downlink bandwidth effectively.* This is achieved by periodically broadcasting the heavily-demanded items. Periodic broadcast of frequently demanded items is very likely to satisfy a large number of users via a single transmission. The remaining items have to be requested and are broadcast on demand.
- *Capability of handling deadline requirements adequately.* A hybrid model can be viewed as having two separate *logical* servers, where the first one prioritizes data items based on their relative popularity, resulting in an efficient usage of the bandwidth, and the second one prioritizes data items based on their relative deadline constraints, trying to send the information to the users by the time that they need it.

Now we proceed to state the three main issues to address when designing an adaptive hybrid broadcast model to work in a dynamic time-critical asymmetric communication environment:

- *What should be the fraction of the total available bandwidth allocated to each broadcast mode?* The amount of bandwidth assigned to each logical server should be adjusted dynamically, depending on the actual user access frequency distribution. There is a trade-off between the relative importance given to popularity and to deadline constraints.
- *Which data items should be broadcast periodically and which ones should be broadcast only when requested explicitly?* Intuitively, an item should be broadcast periodically when there is a potential bandwidth saving with respect to broadcasting it on-demand.
- *How should the periodic and on-demand broadcast programs be computed?* Once we know which items are worthwhile broadcasting periodically, we have to decide their relative broadcast frequencies and compute the periodic broadcast program accordingly. Similarly, on-demand broadcast schedule needs to be constructed.

The adaptive hybrid broadcast model presented in this paper addresses all these issues. The periodic broadcast program is dynamically computed on a per-cycle basis, and the server includes in it only the items that are likely to produce a bandwidth saving when broadcasting them periodically. The items periodically broadcast and their broadcast frequencies (and consequently the amount of bandwidth left for the on-demand mode) depend on the actual access frequency distribution and deadline constraints associated with data, so the amount of bandwidth assigned to each mode changes to adapt to the actual user access pattern, trying to minimize the total number of deadlines missed. Indexing of the periodic broadcast program is used in order to avoid a large number of requests coming to the server. As the periodic broadcast program is computed dynamically, trying to address the needs of most users, the number of requests coming via the uplink channel is greatly reduced and its congestion is much less likely.

3 Related Work

In the traditional Broadcast Disks (BD) model [1], the server periodically repeats an off-line computed broadcast program, based on previously known user access patterns. A *broadcast cycle* is defined as one transmission of the periodic broadcast program. The broadcast program is static and only one downlink channel is used. The main disadvantages of this approach are the lack of adaptation to the current population demands, as there is no feedback from clients to the server, the impossibility of guaranteeing the meeting of deadline constraints for current clients and the potential waste of bandwidth derived from the off-line decision about the items to broadcast and their corresponding broadcast frequency. The BD model has been extended in [2], providing push and pull for data dissemination, by integrating an uplink channel. Nevertheless, in this hybrid model the bandwidth is assigned to the two broadcast modes *statically*, and there is no adaptation to the current workload, as the contents of the periodic broadcast do not change with time. In addition, data requests are not associated with deadlines.

In [15, 16], a model for adapting the allocation of the bandwidth to the periodic and on-demand modes is proposed. This work tries to provide adaptation in order to minimize *average* data access latency, so they minimize the number of requests arriving at the server by broadcasting periodically the most demanded items, providing the rest of the items only on demand. Clients listen to the broadcast first, and make a request only if the data item needed is not in the periodic broadcast program. All requests are served sooner or later, but again, information requests are not associated with deadline constraints. Similarly, in [12], an adaptive algorithm that statistically selects data to be broadcast based on user profiles is presented. Items are added to or dropped from the periodic broadcast based on their relative popularity, how long requests for them have been ignored, and the expected time that a client requesting an item will stay in the domain of the server.

Deadline constraints have been integrated into the BD model in [6, 7, 9]. The server tries to compute a periodic schedule that provides worst-case guarantees, even in the event of failures and data updates. However, this model is not bidirectional, that is, there is no uplink channel and consequently the server periodically broadcasts items based on a *static* estimation of the potential user population, not on the actual workload.

In [10], a reliable multicast protocol is proposed, incorporating a special encoding-decoding scheme to make an effective use of the channel in the face of high packet loss and corruption rates. This model is appropriate for applications that must reliably distribute bulk data to a large number of clients over lossy channels, but it is not bidirectional, that is, contents of a digital fountain are not decided dynamically by clients, as they are not allowed to make requests. Another protocol that integrates loss probability is analyzed in [3]. In this case, as in [16], timing constraints are not considered, and the objective is the minimization of the total average latency.

Closer to our work is the hybrid Broadcast On-Demand (BoD) model, presented in [17], as an extension of the traditional Broadcast Disk model. BoD integrates two separate channels, one high-capacity channel for broadcasting from the server to users and one low-capacity channel that allows the users to transmit requests to the server. A request is associated with a unique identifier, corresponding to the information requested, and a deadline, the maximum tolerable latency to receive the information. BoD combines periodic broadcasts with on-demand data dissemination. Time division multiplexing is used at the server, using a *fixed* fraction of the available bandwidth for periodic broadcast, and the remaining bandwidth for on-demand processing.

One important parameter in hybrid models is the total broadcast cycle length, the global period of the broadcast. The total broadcast cycle length for cycle number i is the sum of the lengths of one cycle of the periodic broadcast and the length corresponding to the bandwidth assigned to on-demand processing for that cycle:

$$Broadcast\_Length_{cycle_i} = Periodic\_Broadcast\_Length_{cycle_i} + On\_Demand\_Length_{cycle_i}$$

In non-adaptive models (e.g., BoD), $Broadcast\_Length_{cycle_i}$ is the same for all i and depends on the number of items being periodically broadcast, their relative broadcast frequencies and the periodic/on-demand bandwidth ratio. This is because, the BoD model assumes that the server is cognizant of the data access distribution in advance. According to this information and using the fixed fraction of the bandwidth available for periodic broadcast, the server decides a priori which items to broadcast (these will typically be the most frequently accessed ones) and at what rate, creating an off-line fixed layout in which the most frequently accessed items are broadcast more often, and copies of the same data item are placed evenly separated, trying to minimize the number of deadlines missed. This precomputed layout is periodically broadcast using the fraction of the bandwidth corresponding to the periodic broadcast mode, and therefore the periodic cycle length and the total broadcast length do not vary with time. BoD is *not adaptive* to dynamic workloads, as it is assumed that the access distribution will remain constant. So, in BD and BoD, a large amount of bandwidth is likely to be wasted, not only because the periodic broadcast program is static and the off-line computed access patterns can differ from actual ones, but also because the broadcast frequency is not related to the deadline distribution of the items.

In our time-critical adaptive hybrid data dissemination model, the objective is to broadcast a given item with the minimum frequency necessary to satisfy all potential requests for that item (given that the requests have deadlines and user request patterns vary from time to time). To this end, the adaptive approach taken in our model adjusts $Broadcast\_Length$ from one cycle to the next so as to respond to user request patterns. In fact, (a) the set of data items broadcast periodically in cycle i, and (b) the amount of bandwidth assigned to each transmission mode, more specifically, $Periodic\_Broadcast\_Length_{cycle_i}$ and $On\_Demand\_Length_{cycle_i}$, could differ from cycle i to the next, to adapt to the needs of the clients.

To summarize, in Table 1 we list the features of the different models described above and of our new time-critical adaptive hybrid broadcast model:

	Adaptive	**Timing constraints**	**Bidirectional**
BD [1]	No	No	No
Extended-BD [2]	No	No	Yes
Real Time-BD [5,6,8]	No	Yes	No
Adaptive Broadcast [11, 12]	Yes	No	Yes
BoD [13]	No	Yes	Yes
New Adaptive Hybrid Broadcast	Yes	Yes	Yes

Table 1: Comparison of broadcast models

We can observe that none of the existing models integrates all the characteristics needed to perform effectively in a real-world, dynamic time-critical asymmetric communication environment. The time-critical adaptive hybrid model (last row in table 1) detailed in the next section encompasses all of them, and also considers the important effect of *uplink channel saturation*, something that has not been addressed by any of the other models.

4 Time-Critical Adaptive Hybrid Data Dissemination

In this section we present the time critical adaptive hybrid broadcast (TC-AHB) model and its scheduling approach. We start with an overview, followed by the approach taken to classify the data items for periodic broadcast and on-demand broadcast. This is followed by details of the scheduling algorithms that dynamically compute the periodic broadcast program and process the on-demand broadcast. We finish this section explaining the sampling technique used by the server to estimate the actual user access frequency distribution.

4.1 Overview

In TC-AHB, both (a) the specific data items being broadcast periodically, and (b) the amount of bandwidth assigned to each transmission mode change dynamically to adapt to the needs of the clients. The server performs this adaptation on a per-cycle basis, in order to use the available bandwidth as efficiently as possible and consequently minimize the total number of unsatisfied users. At the end of each broadcast cycle, the server classifies the data items, deciding which ones to broadcast periodically in the next broadcast cycle. This decision depends on the access distributions observed in the previous broadcast cycle and the amount of bandwidth needed to broadcast each item periodically. As we will see shortly, this amount is related to the deadline constraints associated with each item.

Then, the server computes the periodic broadcast program for the next cycle, leaving some bandwidth for the on-demand broadcast, that is computed dynamically. An item is included in the periodic broadcast program if the server expects to save some bandwidth as compared to broadcasting the item upon request, assuming that there is bandwidth available. We ensure that there is always some minimum fraction of bandwidth left for the on-demand broadcast mode, in order to prevent the items not included in the periodic broadcast program from starving. The broadcast frequency of a periodically broadcast item should be the minimum needed to satisfy all the deadlines associated with potential requests for that item. After computing the periodic broadcast program according to the classification of the items and their broadcast frequencies, the server assigns the remaining bandwidth to on-demand broadcast. In this broadcast mode, an on-line scheduling algorithm is used to prioritize requests according to their relative deadlines and consequently minimize the number of unsatisfied requests coming via the uplink channel. Note that the periodic broadcast program is computed at the end of each cycle, whereas the on-demand broadcast is scheduled dynamically, as requests are received from the users.

The scalability of any hybrid broadcast model is compromised by the number of requests that have to be scheduled by the server and the capacity of the uplink channel. Ideally, we want only a very small percentage of the users to issue requests via the uplink channel, due to the following reasons:

- The capacity of the uplink channel is very limited, because of the inherent asymmetry in the communication environment. In this sense, we can model the channel saturation as a percentage of requests that fail to come through the uplink channel. In contention-based protocols, beyond a certain request rate, the channel becomes congested and the number of requests not arriving successfully at the server increases exponentially, due to a high probability of collisions, decreasing the effective capacity of the uplink channel. For example, if the uplink capacity is 30 Kbps and every request has, for example, 150 bits, only 200 requests/sec. can be accommodated in the uplink channel. For rates above 200 requests/sec., a percentage of requests will be unsuccessful, due to channel saturation. Consequently, the goal is to avoid a large number of requests coming to the server, using the periodic broadcast program to satisfy as many users as possible.
- Every request that arrives at the server has to be scheduled, this is, it has to be inserted in a priority queue. The request insertion cost is $O(logN)$ per scheduled request, where N is the number of requests in the queue. This cost can be important if the server's computing power is not high enough. In the interval of time in which a page is being broadcast, the server has to first look in the current schedule to find the next page to be broadcast, and then access the information corresponding to that particular page, also scheduling all the new requests that remain to be scheduled. Beyond a certain number of requests to be

scheduled, the server simply becomes overwhelmed and cannot complete the scheduling of all the requests. In addition, the length of the request queue is limited. If too many requests arrive, some of them will be just discarded and the number of unsatisfied requests will also increase.

Assuming that some indexing technique is used to help users know what is being broadcast and when, every user is supposed to first listen to the broadcast, check the index, and only issue a request if the desired item is not being broadcast periodically. In our model the index is interleaved with data and it is broadcast many times in each broadcast cycle, in order to decrease the time needed for a client to receive the index. There is a trade-off between the overhead caused by the transmission frequency of the index and the time that users have to wait until they receive a copy of the index. There are several bandwidth efficient indexing techniques (e.g., [13]) that keep the index bandwidth requirements very low. We assume that the index transmission does not incur considerable overhead, and the delay experienced by a user waiting for the index is received is also small.

Due to the fact that a large number of users should be satisfied by the periodic broadcast program and should not issue requests, it is impossible for the server to know the exact user access distribution. The more accurate the periodic broadcast program, the less information the server has about the user access distribution. The problem is that the server does not receive any information about the number of users that are being satisfied by the periodic broadcast program, and therefore is unable to know if any of the items should be removed from the program because its access frequency has decreased. As we will see in section 4.4, this problem is overcome using a sampling technique, that basically stops broadcasting some of the items included in the periodic broadcast during short time intervals. This will result in the receipt of requests for those items. The server can then estimate their actual access frequencies for these requests information needed to dynamically adjust the periodic broadcast program to the needs of the actual users.

4.2 Classification of Data for Periodic and On-Demand Broadcast

The TC-AHB server considers an item to be in one of two states at any given time:

- **Periodic**: included in the current periodic broadcast program
- **On-Demand**: not included in the current periodic broadcast program

At the end of each broadcast period, the server updates the status of every item, depending on the estimated access frequency distribution and the bandwidth required to include it in the periodic broadcast program. Intuitively, items with high access frequency and low bandwidth requirements should be broadcast periodically in order to minimize the number of unsatisfied requests. However, as the bandwidth is limited, the server may not be able to broadcast periodically all items whose access frequency is high and bandwidth requirements are low. The problem is how to determine the allocation of bandwidth for periodic and on-demand broadcast. This should depend on the total workload of the server, that is, under low workload situations, items with relatively low access frequency and high bandwidth requirements should be disseminated

On-Demand (pull-based dissemination), whereas in overload conditions a large number of items should be Periodic (push-based dissemination), in order to satisfy many requests via a single transmission.

Our solution is fairly simple: the server should include a given item in the periodic broadcast when it is likely to save bandwidth by doing so, as long as there is still some predetermined minimum amount of bandwidth left for on-demand broadcast. In order to know if there exists a potential bandwidth saving when classifying an item as Periodic, the server needs to compute first the bandwidth requirements of that item. This depends on its broadcast frequency. In section 4.2.2, we show how to calculate the bandwidth requirements of an item when its broadcast frequency is the minimum needed to satisfy all the deadlines associated with potential requests for that item. First, in 4.2.1, we detail the approach taken to select the specific items that are bandwidth, in order to prevent starvation of the items broadcast on-demand and to keep some bandwidth for transmitting the index. For example, the BW_Threshold would be 0.9 if we want to reserve at least 10% of the available bandwidth for the on-demand mode and the index in the worst case.

The greedy strategy for solving the 0-1 knapsack problem consists on taking first the item with the greatest "value per pound", then the second one, etc. until there is no space left in the knapsack for another item. Unfortunately, this does not guarantee an optimal solution, but if the weights are small compared with the total capacity of the knapsack, the solution is nearly optimal. Fortunately, the solution described below will be optimal in almost every situation, because as we will see shortly, we have an additional condition that stops adding items when there are no more items worth broadcasting periodically. Note that the amount of bandwidth devoted to Periodic (and consequently On-Demand) broadcast adapt to different conditions. While in all cases the amount of bandwidth taken by the periodic broadcast program is *less than or equal to* BW_Threshold, the upper limit, in most situations it is *less than* BW_Threshold.

Only in strong overload conditions does the server use all the available bandwidth for the periodic broadcast schedule, and in that case the number of items periodically broadcast increases considerably. In this case the chances of coming up with an optimal solution are much higher.

Thus, for each data item we could define its priority as the number of requests received for that item during the last broadcast period *divided* by the bandwidth required to broadcast that item periodically. This is equivalent to giving more priority to the items that are likely to satisfy more requests per *bandwidth unit*.

We are now ready to specify the adaptation algorithm executed by the server at the end of each broadcast period:

1. Calculate the bandwidth requirements of an item. Suppose every item has a certain size in pages, or units of data to be broadcast. If we have n items to be scheduled periodically, with sizes $s_1, s_2, \ldots, s_n$ and the relative deadlines of their associated requests $d_1, d_2, \ldots, d_n$. For each (s_i, d_i), its bandwidth requirements are defined as follows:

$$BW\_required_i = \frac{s_i + 1}{d_i + 1}$$

.

2. For each data item, calculate its estimated priority, that is, the number of deadlines expected to be met per bandwidth unit if included in the periodic broadcast program.

$$Priority_i = \frac{Requests\_Received_i}{BW\_Required_i}$$

   Rank all items according to their priorities. The highest priority item is the one with the maximum expected number of deadlines to be satisfied per bandwidth unit.
3. Consider the items according to their priorities, starting with the highest priority item, for inclusion in the periodic broadcast. For each item, proceed as follows:
   (a) Check if it is worth broadcasting the item, that is, if the number of requests received for the item during the last cycle exceeds the *Cutoff_Threshold.* The Cutoff_Threshold is the number of times that we would broadcast the item in a broadcast cycle if the item was classified as Periodic, and equals the last broadcast cycle length divided by the deadline of the item. The rationale is that if we expect to receive less requests than the number of copies of the item that we are going to lay out, then we would waste bandwidth if we consider it Periodic.
   (b) If the item is worth broadcasting, check if the total bandwidth required so far is still below the BW_Threshold when we add this item to the periodic broadcast program. If the total bandwidth required remains below the BW_Threshold, the status of the item is considered to be Periodic for the following period, and then we go back to Step 2a, where the next item is checked. If the total bandwidth required exceeds the BW_Threshold, proceed to step 3.

   To summarize, which items should be Periodic in the next broadcast period? The m highest priority items, given that these conditions hold:

$$\sum_{i=1}^{m} BW\_Required_i \leq BW\_Threshold \quad (1)$$

$$Requests\_Received_i > Cutoff\_Threshold_i \qquad 1 \leq i \leq m \quad (2)$$

4. In step 2, we have used a certain portion of the bandwidth for periodic broadcast, that does not exceed BW_Threshold. The rest of the bandwidth is used to satisfy the requests received via the uplink channel, as detailed in section 4.4.

4.2.1 Computation of the Periodic Broadcast Program

Assuming the server knows a priori the deadline distribution for every item (or, relaxing this assumption, we can assume that the minimum deadline value is known for each item, which is possible if the clients are only allowed to request information using a set of predetermined deadline values), as a first approach we could broadcast a given item with a period equal to the value of the minimum deadline. However, this does not guarantee that all deadlines are met for that item, because the difference between two consecutive broadcasts of the item in the worst case is twice the deadline minus the size of the item. Periodic layouts only guarantee that a given item is scheduled once per period, but in the worst case it can be scheduled once at the beginning of a period and the next time at the end of the period. It would be convenient to compute a schedule that guarantees that all the deadlines will be met for every item included in the periodic broadcast program. As we show below, this is possible if the items are scheduled in a *pfair* manner [4, 5, 8].

There exist algorithms that, if the total bandwidth required is below some threshold (that depends on the specific algorithm used), are able to efficiently compute a periodic schedule in which it is guaranteed that any subsequence of d_i pages contains at least s_i pages corresponding to item i. This periodic schedule is said to be a *pfair* schedule, because it provides fairness and proportionate progress in the allocation of the resource (downlink channel) to the set of items considered.

The scheduling problem we want to solve can be formally stated as follows:

Given a multiset $\{(s_1, d_1), (s_2, d_2), \ldots, (s_n, d_n)\}$ of ordered pairs of positive integers, determine a sequence over the symbols $1, 2, 3, \ldots, n$ such that, repeating the sequence periodically, for each i, $1 \leq i \leq n$ any subsequence of d_i consecutive symbols contains at least s_i i's.

The algorithm described below can be used to generate the desired sequence. It is based on the *PinOpt*[8] algorithm. First, we need to define the following:

- $allocated(i, t)$ = number of slots allocated to item i in the interval $[0, t)$
- Item i is said to be *contending* at time slot t if it may receive the resource without becoming overallocated, that is, if the following condition is true:

$$\left\lceil \frac{allocated(i,t)}{BW\_Required_i} \right\rceil \leq t$$

- The pseudodeadline of item i at time slot t is defined as:

$$pseudodeadline(i,t) = \left\lceil \frac{allocated(i,t)+1}{BW\_Required_i} \right\rceil$$

These are the two steps that form the algorithm:

- **Step 1**: if $\sum_{i=1}^{n} BW\_Required_i > 1$, return failure, else if $\sum_{i=1}^{n} BW\_Required_i < 1$, add a dummy pair (s_{n+1}, d_{n+1}) such that $\sum_{i=1}^{n+1} BW\_Required_i = 1$. The slots corresponding to item $n+1$ will be used to broadcast copies of the index and the items served on-demand.

- **Step 2**: schedule the new multiset using the following algorithm: assign each broadcast slot to the *contending* item with the smallest pseudodeadline. Ties can be broken arbitrarily.

Intuitively, the resulting schedule is fair in the sense that it guarantees that each item i between 1 and $n+1$ is scheduled exactly once during the interval $[\lceil \frac{j}{BW\_Required_i} \rceil, \lceil \frac{j+1}{BW\_Required_i} \rceil)$, for each integer $j \geq 0$, assuming that the condition $\sum_{i=1}^{n+1} BW\_Required_i = 1$ holds.

This algorithm can be efficiently implemented using a heap-of-heaps data structure, in $O(logn)$ per time slot [14]. The period of the schedule generated is the least common multiple of the deadlines, and this will be the broadcast cycle length. This is another good reason to restrict the values of the deadlines to a predefined small set, to keep the broadcast length values within a reasonable range.

The periodic broadcast program is laid out using the algorithm just described. Consequently, each item i broadcast periodically will have s_i pages broadcast every d_i pages, and all deadline requirements for these items will be automatically satisfied by the periodic broadcast, because a given client that needs item i is guaranteed to receive s_i pages in a period of time not greater than d_i, the minimum deadline associated with that item. It is assumed that clients can reorder data pages locally. As an example of the layout computed by the algorithm, consider items i_1, i_2 and i_3, characterized by $\{(3,5),(1,3),(1,15)\}$, as the input. The periodic broadcast program generated is the following:

slot:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
item:	i_1	i_2	i_1	i_2	i_1	i_1	i_2	i_1	i_3	i_1	i_1	i_2	i_1	i_2	i_1

4.3 Scheduling the On-Demand Broadcast

When a request arrives at the server, it means that the periodic broadcast will not satisfy it, and therefore the on-demand broadcast has to be used to handle it. Requests should correspond to relatively infrequently demanded information or data with high bandwidth requirements, which is not part of the periodic broadcast program.

The on-demand broadcast mode requires the use of an on-line server transmission scheduling policy. In order to take into account deadline requirements, a scheduling policy that tries to minimize the number of deadlines missed is used, specifically the Earliest Deadline First policy. We should note that EDF is optimal only if all the items are equally sized; if this is not the case, there are other scheduling policies that may perform better than EDF. Requests are inserted in a priority queue, where its priority is inversely proportional to the relative deadlines of the requests. The schedule corresponding to the on-demand broadcast does not take into account the relative access frequency of the data, only the relative deadlines. Requests will be satisfied by the server according to their priorities, using the bandwidth remaining for the on-demand broadcast to transmit the required data items, in the slots reserved to the dummy item $n+1$ by the algorithm described in the previous section. The specific amount of bandwidth dedicated to on-demand processing depends on the length of the periodic broadcast program, and therefore will change according to the actual user access pattern. However, as pointed out in 4.2.1, there is always a minimum amount of bandwidth reserved for on-demand broadcast and the broadcast of the index, namely, (1-BW_Threshold).

The performance of the EDF scheduling policy is increased by batching multiple requests for the same data item, satisfying all the batched requests via a single transmission of the requested data item. When a request for a given data item comes via the uplink channel, the server first checks if there is already a transmission scheduled for the same item with a deadline not greater than the one associated with the request, in which case the scheduled transmission will satisfy it, and consequently is not necessary to schedule another transmission. This batching technique produces an important saving of bandwidth in overload conditions, resulting in a large increase of satisfied users, as shown in [17].

4.4 Sampling: Estimating the Access Frequency Distribution

The transmission of a periodic broadcast program combined with indexing decreases the number of requests arriving at the server, clearly increasing the scalability of the server. Unfortunately, this causes the server not to receive any information about the number of requests corresponding to Periodic items, information the server needs to calculate the priority of these items in subsequent broadcast periods, as seen in 4.2.1.

The problem is the following: assuming that a given item has been requested a large number of times during the previous cycle, and it is included in the periodic broadcast program, the server does not receive any request for that item in the present cycle, and consequently the server is unable to estimate the real number of users that were satisfied by broadcasting the item periodically during the cycle. To overcome this problem, we have incorporated a sampling technique into the model, inspired by the method used in [16]. When an item is added to the periodic program, the number of requests received in the previous cycle is conveniently stored for later use. At the end of each subsequent cycle, this number is artificially decreased (it is multiplied by a *Cooling Factor*, typically in the range 0.7-0.9), and this *fictitious* number of requests received is used to calculate the estimated priority of the item.

After a few cycles, this estimated priority decreases to the point in which the item would not be included in the periodic broadcast. At that point, before removing the item from the periodic broadcast program, the sampling technique is used to estimate the access frequency for the item during the next broadcast period. In the following broadcast cycle, the server still includes the item in the periodic broadcast, but only for part of the cycle. For a short interval at the end of the cycle, the server decides not to broadcast the item, in order to receive a few requests and obtain information about the present access frequency of that item.

The index, interleaved with the data, is updated to reflect this change. This forces users to make explicit requests for these items, whereby the server collects a small number of requests for that item, and using this small number, it estimates the total number of requests that would have been received during the whole cycle. This estimation is used to compute the actual priority of the item, and finally decide if its status should remain Periodic or changed to On-Demand. This technique allows a Periodic item to remain Periodic if appropriate.

The time interval in which the server collects the requests is a crucial design parameter, and as suggested in [16], its value should be calculated so that only a few requests are received, in order for the sampling to be effective, otherwise this technique essentially defeats its purpose. This small number of requests that the server expects to receive when sampling an item is called the *Expected_Sample_Size*. The time needed to get the sample is easy to compute, as we know how many requests were received for item i during the broadcast cycle prior to its inclusion in the periodic broadcast program (cycle j), and therefore we can calculate the amount of time needed to receive Expected_Sample_Size requests:

$$Sampling\_Time_i = \frac{Broadcast\_Cycle\_Length_{cycle_j} \times Expected\_Sample\_Size}{Requests\_Received_{i,cycle_j}}$$

Finally, the estimated number of requests received for i during the cycle in which it is sampled (cycle k) can be calculated once the server knows the actual sample size:

$$Estimated\_Requests_{i,cycle_k} = \frac{Broadcast\_Cycle\_Length_{cycle_k} \times Actual\_Sample\_Size_i}{Sampling\_Time_i}$$

This estimation is used to compute the priority of the item, allowing the item to remain in the periodic broadcast program if the estimated number of requests is still high enough.

5 Experiments and Results

In order to evaluate quantitatively the performance of our time-critical adaptive hybrid model, we have carried out several simulation-based experiments. The clients are modeled by a workload generator, that produces an effective information demand rate. The interarrival time of client demands is exponentially distributed, that is, the sequence of demands is a Poisson process. Some of these demands will be satisfied by the periodic broadcast and so will not result in requests sent to the server. The smaller the number of requests sent to the server, the better the periodic broadcast program.

Given this, we have two metrics to evaluate the goodness of an algorithm designed for data dissemination:

1. The total percentage of information demands satisfied, i.e., the percentage of cases where a user is able to obtain the needed information (from the periodic broadcast or the on-demand broadcast) before the deadline associated with the demand. This serves as our primary performance metric. In the graphs we portray the performance in terms of the complement of this metric by showing the percentage of deadlines (associated with information demands) that are missed. Note that it is incorrect to use the percentage of satisfied *requests* as a performance indicator, because many of the user needs are likely to be satisfied by the periodic broadcast program, i.e., without a request being sent to the server.

2. The total number of requests that are sent to the server (corresponding to user demands not satisfied by the periodic broadcast program). This metric helps to evaluate the effect of the uplink channel saturation.

Table 2 shows some important simulation parameters and their corresponding values. The rationale behind the choices is explained below. (These values are very similar to the corresponding ones in [16, 17], and represent a realistic asymmetric communication environment.)

The time unit in our simulator is the time needed to broadcast a single data page (along the downlink). Thus, it is not necessary to specify the downlink channel capacity, because in a real situation this parameter would only affect the time needed to broadcast a single page, and we are taking this amount as our time unit.

Parameter	**Value**
Number of items	1000
Size of each item	10 pages
Relative deadlines	100, 200 or 500 time units
Simulation time	1000000 time units
BW_Threshold	0.9
Cooling factor	0.9
Sample size	5 requests
Uplink capacity	1 request/time unit
Workload range	0.1 to 4 items demanded/time unit

Table 2: Simulation parameters

Two different access frequency distributions are modeled:

- *Two-level Uniform*, combining two different access frequencies in the same distribution. The first 20 items are accessed with 90% probability, the rest of the items only with 10% probability.
- *Zipftian*, where the probability of accessing an item i is proportional to $\frac{1}{i}$.

The experiments were carried out using static and dynamic versions of these workloads. The dynamic uniform workload changes the specific 20 heavily-demanded items four times during the simulation time. Initially, at time 0, the starting range is [1,20] becoming [201, 220], [401, 420], [601, 620], [801, 820] at times 200000, 400000, 600000, 800000. The dynamic zipftian workload is constructed by shifting the static zipftian distribution to the right, that is, adding a small quantity to the number of the item accessed. Initially, at time 0, the quantity is 0, becoming 20, 25, 30, 35 at times 200000, 400000, 600000, 800000. Note that, as the size of the items is 10 pages, the workload ranges from 1 page demanded per time unit to 40 pages demanded per time unit, and consequently the server works in overload conditions over the whole workload range.

The deadline distribution is uniform, allowing only a small set of values {200, 500, 1000}, in time units. For a given item, one of these values is permanently associated with it. All the requests for an item have the timing constraint associated with the item.

As pointed out in [16], there is a trade-off in the values of the cooling factor and the sample size. Clearly, the lower the value of the cooling factor, the faster the adaptation to changes in the user access distribution is. The problem is that a low cooling factor also implies that Periodic items are going to be sampled more frequently, and this increases the number of requests received, increasing the overhead. This overhead is lowered by selecting a very small probing size, but this results in loss of accuracy in the estimation of the requests received during the whole cycle. As our dynamic workloads do not change very frequently, we decided to choose a high cooling factor, 0.9, and a moderately low sample size, 5 requests.

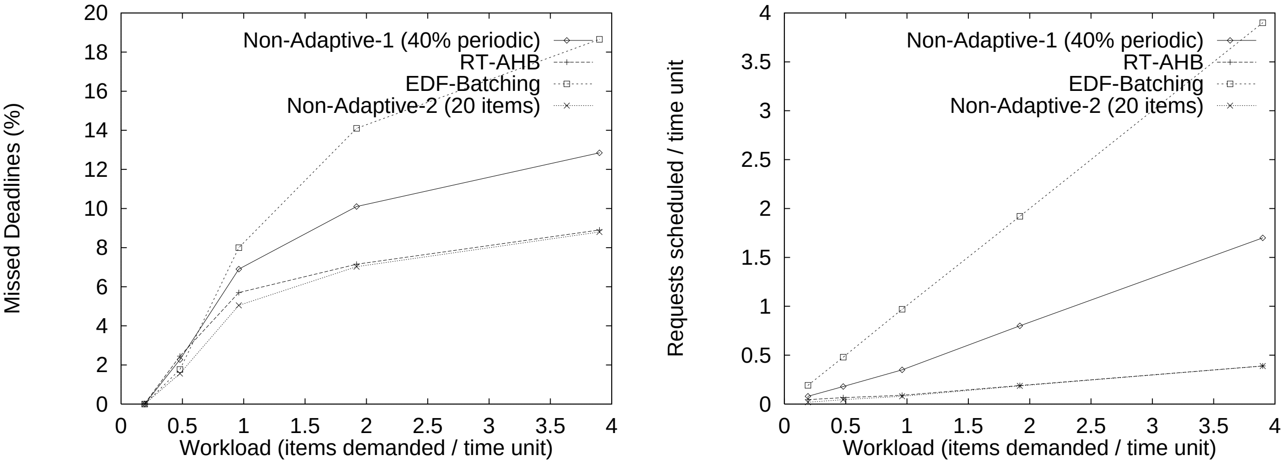
The uplink capacity has been set to one request per time unit, and it can only be achieved when the offered workload is exactly one request per time unit. If the offered workload is higher than one request per time unit, the effective real capacity of the uplink channel decreases, due to an increasing number of collisions, as the uplink protocol is contention based (a CSMA-like protocol has been used). We model the effect of contention by assuming that, after the maximum capacity is reached, the capacity of the channel decreases linearly as the workload increases.

As mentioned in the third section, [16] does not associate requests with deadlines, and there is only one hybrid (but non-adaptive) model in which temporal constraints are taken into account [17]. So, we compare TC-AHB with two protocols based on this non-adaptive model. This comparison will give us an idea of the performance improvement resulting from the adaptation of TC-AHB to the actual user access distribution with dynamic workloads, as the periodic broadcast program is computed on-line and changes according to the user access distribution during the simulations. We also include a non-hybrid protocol (there is no periodic broadcast program) in the comparison. This model has been included in order to measure the benefits of TC-AHB derived from the effective use of uplink and downlink channels.

The difference between the two non-adaptive protocols is that the first one uses a fixed fraction of the bandwidth for the periodic broadcast program, whereas the second one broadcasts periodically the first 20 items, and is only used with the two-level uniform distribution. The reason for this is that, in the two-level uniform distribution, we want to compare our model with (a) a non-adaptive model in which the periodic broadcast program occupies a fixed fraction of the bandwidth, and (b) with a non-adaptive model in which the most frequently demanded items are exactly the ones included in the periodic broadcast program.

A brief description of the three models that are compared to TC-AHB follows:

- *Non-Adaptive-1.* This is a non-adaptive hybrid model similar to BoD, in which a fixed amount of bandwidth is allocated for the periodic broadcast program. The items broadcast periodically are decided off-line. In this case, the first i items (1, 2, 3, etc.) that fit in a fixed fraction of the available bandwidth are included in the periodic broadcast mode. The broadcast program is static, and the periodic broadcast program is computed off-line, in a *pfair* manner. This model uses the same scheduling policy as our adaptive model for the on-demand broadcast mode, EDF with batching. When the distribution is zipftian, the fraction of bandwidth allocated to the periodic broadcast is 50%, as in BoD.



(a) Missed Deadlines (b)Requests Scheduled

Figure 1: Static Uniform Distribution, Unlimited Uplink Channel.

In the two-level uniform case, we decreased the amount of bandwidth allocated to the periodic broadcast to 40%, so that some of the first 20 items were not present in the periodic broadcast program. This allows us to measure the effect of a slight mismatch between the periodic broadcast program and the most heavily-demanded items.

- *Non-Adaptive-2*: This model is only used with the two-level uniform distribution. The only difference from the Non-Adaptive-1 is that now the periodic broadcast program includes the first 20 items, instead of using a predetermined fraction of the bandwidth.
- *EDF-Batching*. This is a pure pull-based broadcast model, not cycle-based. It uses an on-line EDF scheduling policy with batching. We can think of this model as a particular case of a hybrid model, where no items are broadcast periodically.

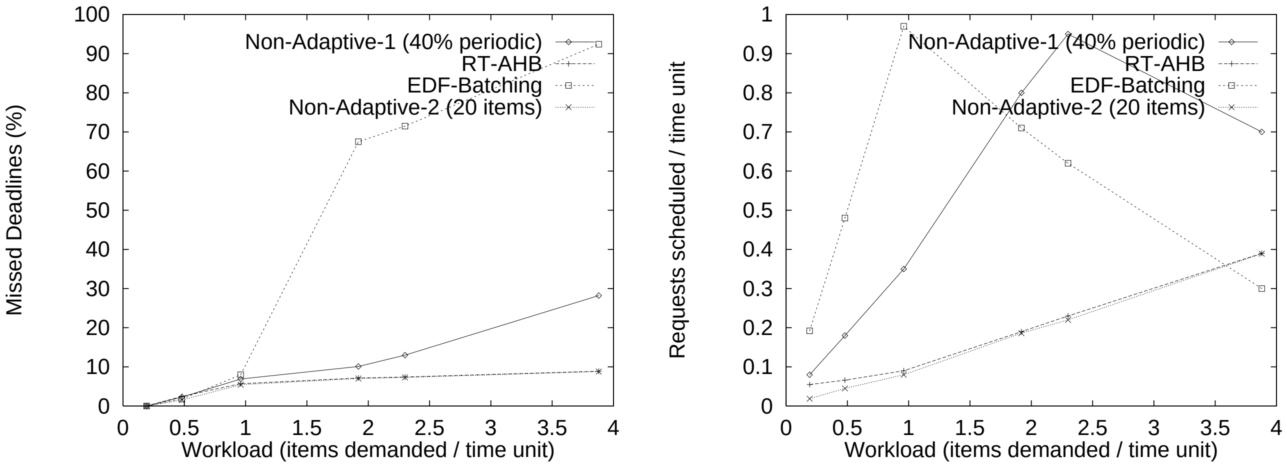
The index is interleaved with the data and its bandwidth requirements are equivalent to 1% of the total bandwidth. The index is transmitted every 10 time units and occupies 1/10 of a time unit (the size of the index is 1/10 of a single data page). The index waiting-time overhead has been taken into consideration in the experiments.

Each point obtained in the simulations is the average of 5 different simulation runs. In the worst case, the 95% confidence interval equals 8.2% of the mean when measuring the percentage of deadlines missed, and 4.3% when measuring the number of requests scheduled per time unit.

5.1 Static Client Profiles, Unlimited Uplink Channel

Our first experiment uses the static two-level uniform distribution as the user access distribution, with unlimited capacity for the uplink channel. This is clearly an unrealistic scenario, but it is a good starting point for comparison purposes, as the subsequent experiments use dynamic client profiles and a low-capacity uplink channel.

As we observe in figures 1(a) and 1(b), which plot the percentage of missed deadlines and the number of requests scheduled per time unit respectively, the curves for Non-Adaptive-2 and



(a) Missed Deadlines (b) Requests Scheduled

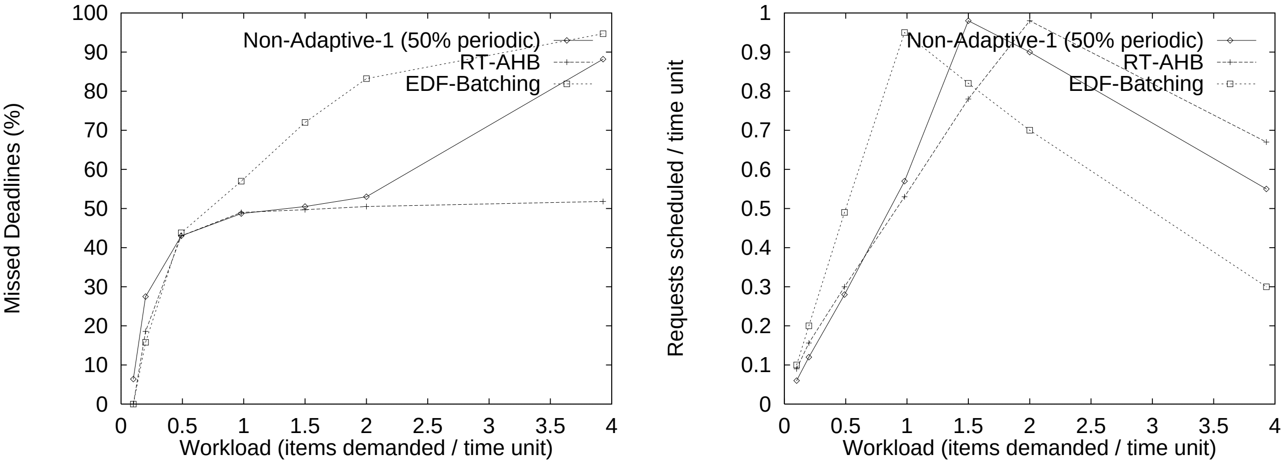
Figure 2: Static Uniform Distribution, Limited Uplink Channel

TC-AHB are almost identical. For low workloads, TC-AHB does slightly worse because of the overhead caused by the sampling technique used in TC-AHB, as the number of requests needed to include an item in the periodic broadcast program is small and consequently the sampling time becomes a bigger fraction of the broadcast cycle length. For example, suppose that 10 requests were received for a given item and at that point it became part of the periodic broadcast. The time interval needed to receive 5 (Sample_Size) requests is equal to half broadcast cycle. This means that, when the server needs to get a sample for that item, during half broadcast cycle the server has to stop broadcasting it and schedule all the requests corresponding to that item.

Therefore, the number of deadlines missed and requests scheduled in the TC-AHB server are slightly higher than the ones in the Non-Adaptive-2 server for low workloads.

In the Non-Adaptive-1 server, since 40% of the bandwidth is assigned to the periodic broadcast program some (4, 5 or 6, depending on the simulation run) of the 20 high access frequency items are not broadcast periodically. Figures 1(a) and 1(b) show that the number of deadlines missed and requests scheduled are higher than the Non-Adaptive-2 and TC-AHB models, and the difference becomes larger as the workload increases. This gives us an idea of the penalty paid when the static broadcast program does not exactly match the actual user access distribution. The fact that some of the heavily demanded items are not included in the periodic broadcast program results in more requests to be scheduled and more items to be broadcast on-demand, increasing the percentage of missed deadlines.

For very low workloads, the performance of the EDF-Batch server is close to Non-Adaptive-2 and slightly better than Non-Adaptive-1 and TC-AHB, because there is no periodic broadcast and no bandwidth is wasted. Nevertheless, EDF-Batch misses more deadlines than the other three models as workload increases, due to the waste of bandwidth resulting from ignoring the relative access frequency of the data. Although batching allows us to satisfy several clients via a single transmission, under strong overload conditions, in conjunction with EDF it is not as



(a) Missed Deadlines (b) Requests Scheduled

Figure 3: Static Zipf Distribution, Limited Uplink Channel

effective as the approach of the hybrid models, which take into account the user access pattern to broadcast the most demanded items periodically.

In addition, as shown in figure 1(b), the number of requests scheduled in EDF-Batch grows linearly with the workload, because all the clients have to make an explicit request in order to obtain the desired information, as there is no periodic broadcast program. On the contrary, for TC-AHB and Non-Adaptive-2 only roughly 10% of these requests are issued. In Non-Adaptive-1, this number is around 45%, due to the fact that some heavily demanded items are not part of the periodic broadcast.

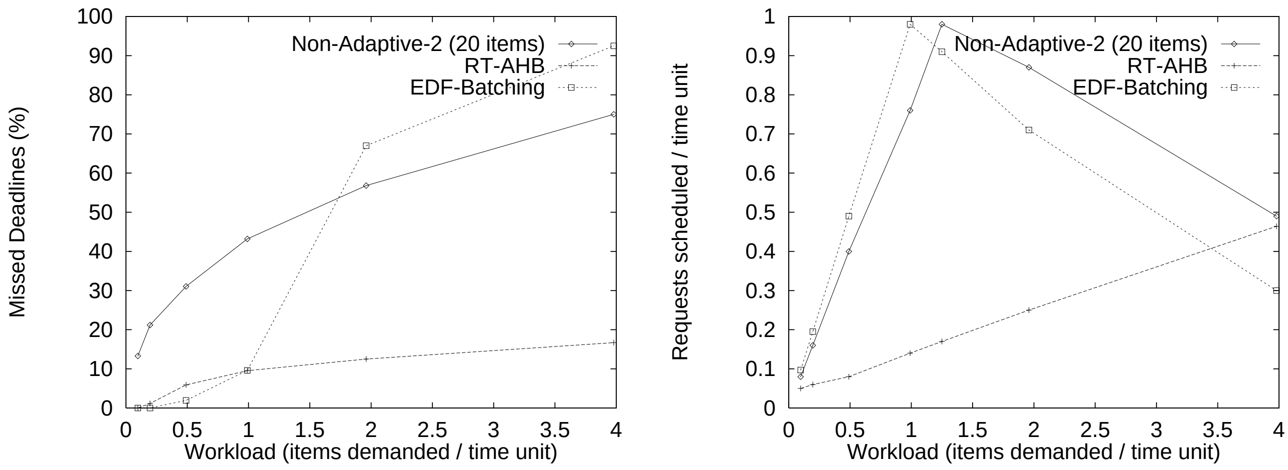
5.2 Static Client Profiles, Limited Uplink Channel

The following experiments measure the effect of the limits in the uplink channel. We should point out that the maximum potential request rate, 4 items demanded per time unit, is four times the uplink channel capacity, which is one request per time unit.

5.2.1 Static Uniform Distribution

Figure 2(a) shows that the limited uplink channel capacity has no effect in the TC-AHB and Non-Adaptive-2 models, as the curves are identical to the corresponding ones in figure 1(a). The reason is that the periodic broadcast program in these two cases fits the needs of the actual user population, obviating the need for a large percentage of the potential requests, and consequently the uplink channel is not congested. The number of requests scheduled per time unit (figure 2(b)) is only near 0.4 even in the worst case.

However, the EDF-batch and Non-Adaptive-1 models do suffer from the effect of uplink channel saturation, and beyond the point in which the workload starts saturating the uplink channel, which is 1 and approximately 2.3 items demanded per time unit (see figure 2(b)), many requests are suffering collisions and not arriving successfully at the server. Due to this congestion effect, the number of deadlines missed increases very fast after the saturation point (see figure



(a) Missed Deadlines (b) Requests Scheduled

Figure 4: Static Zipf Distribution, Limited Uplink Channel

2(a)), and the number of requests that arrive successfully at the server decreases linearly when the offered workload is higher than the point at which the uplink channel gets saturated, as figure 2(b) shows.

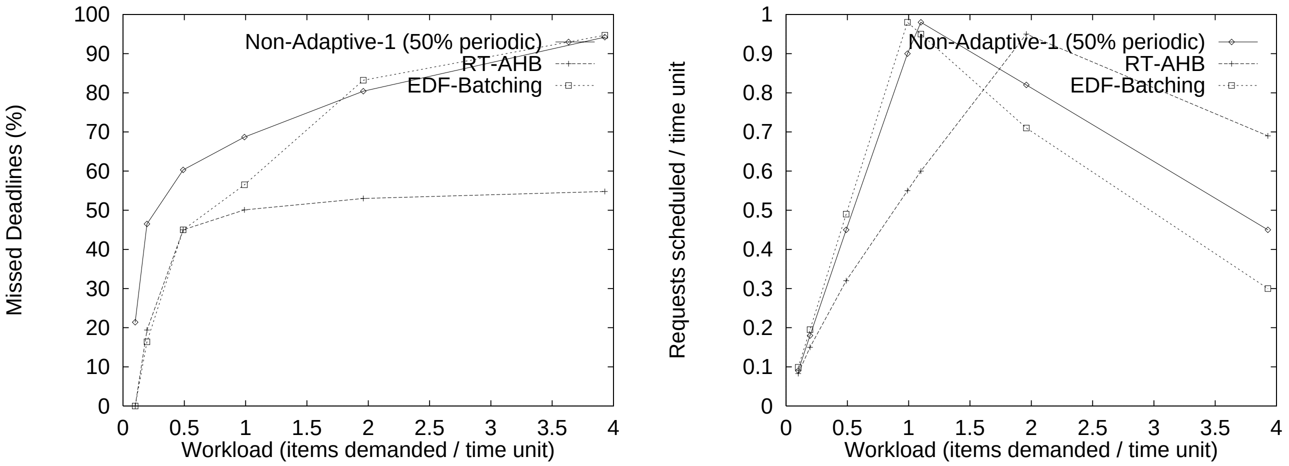
5.2.2 Static Zipf Distribution

Figure 3(a) shows the percentage of deadlines missed for the static Zipf distribution. We can observe that there is a crossover point between EDF-Batch and Non-Adaptive-1, when the workload is around 0.5. The EDF-Batch server performs better than the Non-Adaptive-1 below that point, whereas it does worse as the workload increases.

Again, the reason is that, under low workloads, most items are not worth broadcasting, and the Non-Adaptive-1 model is wasting bandwidth by using 50% for the periodic broadcast program. On the contrary, under high workloads the EDF-Batch is not able to use the bandwidth effectively, because its scheduling approach is only influenced by deadline requirements, not by access frequencies of the data.

The TC-AHB model gets the best of both worlds: it performs similarly to EDF-Batch under low workloads, as very few items are Periodic, and clearly better than both Non-Adaptive-1 and EDF-Batch under high workloads, because it allocates most of the bandwidth to the periodic broadcast (the BW_Threshold chosen implies that at least 10% of the bandwidth has to be reserved for the on-demand mode and the index), broadcasting more items periodically than the Non-Adaptive-1 version and using the bandwidth much more efficiently. In fact, we can see that in TC-AHB the number of deadlines missed does not increase significantly even in strong overload conditions.

Figure 3(b) shows that saturation is felt by the three models compared, but the saturation point is reached earlier in EDF-Batch, when the workload equals one item demanded per time unit. For Non-Adaptive-1, the uplink channel starts to become saturated when the workload is



(a) Missed Deadlines (b) Requests Scheduled

Figure 5: Dynamic Zipf Distribution, Limited Uplink Channel

1.5, and for TC-AHB it happens at 2 items demanded per time unit. This is the reason why the number of deadlines missed grows so fast in Non-Adaptive-1 in strong overload conditions.

In figure 3(b) we can also notice that, before uplink channel saturation, there is also a crossover point: TC-AHB schedules more requests than Non-Adaptive-1 when the workload is in the interval [0.1, 0.7], as the periodic broadcast program is very small, whereas in the interval [0.7, 1.5] TC-AHB schedules less requests, as a larger number of items is included in the periodic broadcast program and consequently the users do not send requests in order to receive them.

5.3 Dynamic Client Profiles, Limited Uplink Channel

In the following experiments, the access frequency distributions do change with time, as in a more realistic situation.

5.3.1 Dynamic Uniform Distribution

As stated before, in the dynamic two-level uniform distribution, the specific 20 items with the highest access frequencies are changed four times during the simulation time. Non-Adaptive-1 has not been included in the comparison, as its performance is very similar to Non-Adaptive-2. The Non-Adaptive-2 model includes the first 20 items in the periodic broadcast program during all the simulation time. That means that Non-Adaptive-2 is periodically broadcasting 20 items that do not correspond to the most frequently demanded ones after the first change in the distribution takes place (t=200000), and therefore it is wasting bandwidth after that and allowing all the requests corresponding to the new heavily-demanded items to be issued.

As a consequence, as we can see in figure 4(a), the number of deadlines missed in the Non-Adaptive-2 model is much higher as compared to TC-AHB, over the whole workload range. The similarity between the TC-AHB curves in figures 2(a) and 4(a) proves that, unlike non-adaptive models, TC-AHB is appropriate for environments where the user profiles change, without a significant loss in performance. There is a small performance difference in TC-AHB between

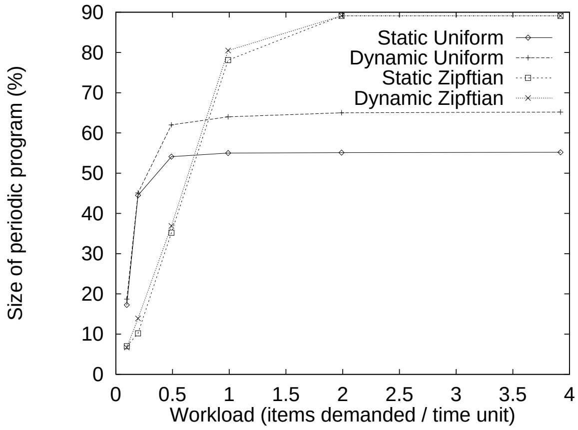


Figure 6: Bandwidth Required for Periodic Broadcast Program in TC-AHB

figure 2(a) (static uniform) and figure 4(a) (dynamic uniform), about 7% in the worst case, when the workload is 4. This is due to the time that it takes to adapt the periodic broadcast program every time the most 20 heavily-demanded items change.

Figure 4(b) shows that, in the Non-Adaptive-2 model, the uplink channel becomes saturated when the workload is approximately 1.2, just after EDF-Batch. The adaptation to the actual user access distribution carried out by TC-AHB makes it possible to keep the uplink channel away from saturation: in the worst case, only about 46% of its capacity is required.

As expected, EDF_Batch obtains the same performance for static and dynamic distributions, as its scheduling algorithm does not take into account the relative access frequencies of the data items. As in the static case, EDF_Batch performs worse than TC-AHB as workload increases in terms of deadlines missed, and also suffers from uplink channel congestion to a much larger extent.

5.3.2 Dynamic Zipf Distribution

When the dynamic Zipf distribution is used, the Non-Adaptive-2 model performs significantly worse than TC-AHB for all workloads. Figure 5(a) shows that the number of deadlines missed by Non-Adaptive-2 is much higher than the equivalent in TC-AHB in all cases. The TC-AHB curve is practically identical to the corresponding one for the static Zipf distribution in figure 3(a). The difference in performance with the static curve is only about 3% in the worst case, which means that TC-AHB is practically insensitive to changes in the access frequency distribution.

In figure 5(b) we notice that Non-Adaptive-2 is only able to avoid a small percentage of requests from being issued, resulting in a very early uplink channel saturation, when the workload is 1.1. TC-AHB avoids the uplink channel saturation for all workloads less than 2, obtaining a good performance in terms of deadlines missed even in strong overload situations.

The comments for the EDF-Batch model are the same made for the dynamic uniform distribution.

5.4 Discussion of Results

Among the different alternatives considered, TC-AHB is the only reasonable choice for obtaining adequate performance in a dynamic time-critical asymmetric communication environment. The TC-AHB model is able to adapt dynamically to different workloads and changes in the access frequency distribution, allowing higher workloads without uplink channel saturation and missing a smaller number of deadlines than the rest of the models. The reason is that the periodic broadcast program is dynamically adapted to the actual users' needs. Figure 6 shows the adaptation of the amount of bandwidth allocated to the periodic broadcast program for the different client profiles. Under low workloads, only a few items are part of the periodic broadcast program, which grows as the workload increases. We can observe that when the uniform distribution is used, the size of the program grows with the workload until the 20 heavily-demanded items are included in the periodic broadcast program, remaining constant after that point, approximately when the workload equals 1. In the zipf distribution, the periodic program keeps adding items as workload increases, until the BW_Threshold is reached, approximately when the workload is 2.

The results show that non-adaptive models compare favorably with TC-AHB only when the client's profiles are static, and even in that unrealistic situation, the fact that the fraction of bandwidth assigned to each broadcast mode is fixed does not allow non-adaptive models to perform well over the whole workload range, unless (a) there is a clear difference between frequently demanded items and non-frequently demanded items (e.g., two-level uniform distribution) and (b) the precomputed broadcast program exactly includes all the frequently demanded items.

It should also be clear that EDF-Batch is inadequate unless the workload is very light, a high-capacity uplink channel is used, and the server has enough resources to schedule a large number of requests. The two main problems of this pure pull-based model are: (1) it is unable to use the bandwidth effectively in overload situations, and (2) all the potential requests have to be issued, causing the uplink channel to become congested earlier than in the hybrid models.

Tables 3 and 4 summarize the performance obtained by the different models compared, for all workloads. The workload range has been divided into 3 parts, corresponding to the left part of the graphs (low overload), middle part (moderate overhead) and right part (strong overhead). The non adaptive models are the only ones sensitive to changes in the access distribution. We observe that TC-AHB is the only model capable of performing adequately over the whole workload range. Moreover, except in low overload situations where it does slightly worse than EDF-Batch, it always obtains the best performance in terms of the number of deadlines missed. In addition, in TC-AHB the uplink channel suffers from saturation only in extreme overload conditions.

	Low overload	**Moderate overload**	**Strong overload**
Non-Adaptive-1	Good/Bad	Poor/Bad	Bad/Bad
Non-Adaptive-2	Best/Bad	Best/Bad	Best/Bad
EDF-Batch	Best	Bad	Bad
TC-AHB	Good	Best	Best

Table 3: Static/Dynamic Uniform Distribution

	Low overload	**Moderate overload**	**Strong overload**
Non-Adaptive-1	Poor/Bad	Close to Best/Bad	Bad/Bad
EDF-Batch	Best	Bad	Bad
TC-AHB	Close to Best	Best	Best

Table 4: Static/Dynamic Zipf Distribution

6 Conclusion and Future Work

In this paper we have presented and evaluated a new model for effective data dissemination in dynamic, time-critical asymmetric communication environments. Our model uses adaptive hybrid broadcast transmission, allowing the users to request specific items with deadline requirements. In order to minimize the total number of deadlines missed by making the most effective use of the available bandwidth, its scheduling approach combines factors such as access frequency, deadline distribution and bandwidth requirements. The results obtained from simulation show that, by broadcasting periodically only those items that are predicted to have high access frequencies and low bandwidth needs, the scheduling algorithm used in the broadcast information server clearly outperforms non-adaptive models when dynamic client profiles occur and in overload conditions.

The dynamic adaptation of the contents of the periodic broadcast program to the real client access profile is essential for any broadcast server to work in environments where the data access distribution changes over time, as in most real situations (specially if the clients have mobility). Simulation results show that our time-critical adaptive hybrid data dissemination model satisfies more clients than other broadcast models, decreasing also the number of requests that have to be issued in heavily overload conditions, which is an important aspect of scalability.

The effect of the bandwidth limitations of the uplink channel is an important factor to be considered. Our adaptive broadcast model assigns more bandwidth to the periodic transmission mode when the workload is high, avoiding an excessive request traffic coming to the server and accommodating higher workloads before the uplink channel becomes congested.

Future work includes extending the functionality of the server by incorporating loss probability, different broadcast channels, and protocols and scheduling algorithms that will allow multimedia information broadcasting and Quality of Service guarantees.

References

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

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

[3] K. Almeroth, M. Ammar and Z. Fei, *"Scalable Delivery of Web Pages Using Cyclic Best-Effort (UDP) Multicast"*, Proceedings of IEEE Infocom '98, San Francisco, CA, March 1998.

[4] S. Baruah, N. Cohen, C. Plaxton and D. Varvel, *"Proportionate Progress: A Notion of Fairness in Resource Allocation"*, Algorithmica 15(6), pgs. 600-625, 1996.

[5] S. Baruah, *"Fairness in periodic real-time scheduling"*, Proceedings of 16th Real-Time Systems Symposium, Pisa, Italy, December 1995.

[6] S. Baruah and A. Bestavros, *"Pinwheel Scheduling for Fault-tolerant Broadcast Disks in Real-Time Database Systems"*, Proceedings of IEEE International Conference on Data Engineering, Birmingham, England, April 1997.

[7] S. Baruah and A. Bestavros, *"Real-Time mutable broadcast disks"*, Proceedings of Second International Workshop on Real-Time Databases, Burlington, VT, September 1997.

[8] S. Baruah and S. Lin, *"Improved scheduling of generalized pinwheel task systems"*, Proceedings of Fourth International Workshop on Real-Time Computer Systems Applications, Teipei, Taiwan, October 1997.

[9] A. Bestavros, *"AIDA-based Real-Time Fault-Tolerant Broadcast Disks"*, Proceedings of Second IEEE Real-Time Technology and Applications Symposium, Boston, MA, June 1996.

[10] J. Byers, M. Luby, M: Mitzenmacher and A. Rege, *"A Digital Fountain Approach to Reliable Distribution of Bulk Data"*, Proceedings of ACM Sigcomm '98, Vancouver, Canada, September 1998.

[11] T. Cormen, C. Leiserson and R. Rivest, *"Introduction to Algorithms"*, McGraw-Hill, 1990.

[12] A. Datta, D. VanderMeer, J. Kim, A. Celik and V. Kumar, *"Adaptive Broadcast Protocols to Support Efficient and Energy Conserving Retrieval from Databases in Mobile Computing Environments"*, Proceedings of 13th International Conference on Data Engineering, Birmingham, U.K., 1997.

[13] T. Imielinski, S. Viswanathan and B. Badrinath, *"Energy Efficient Indexing on Air"*, Proceedings of ACM SIGMOD Conference, May 1994.

[14] A. Mok, *"Task Management Techniques for Enforcing ED scheduling on a Periodic Task Set"*, Proceedings of 5th IEEE Workshop on Real-Time Software and Operating Systems, Washington D.C., May 1988.

[15] K. Stathatos, N. Roussopoulos and J. S. Baras, *"Adaptive Data Broadcasting Using Air-Cache"*, Proceedings of the 1st International Workshop on Satellite-based Information Services, Rye, NY, 1996.

[16] K. Stathatos, N. Roussopoulos and J. S. Baras, *"Adaptive Data Broadcast in Hybrid Networks"*, Proceedings of the 23rd VLDB Conference, Athens, Greece, 1997.

[17] P. Xuan, S. Sen, O. Gonzalez, J. Fernandez and K. Ramamritham, *"Efficient and Timely Dissemination of Data in mobile Environments"*, Proceedings of the Third IEEE Real Time Technology and Applications Symposium, Montreal, Canada, June 1997.