A Cost Model for Scheduling On-demand Data Broadcast in Mixed-type Request Environments

Ming Lei, Susan V. Vrbsky, and Yang Xiao

Department of Computer Science
The University of Alabama
Tuscaloosa, AL 35487-0290
{mlei, vrbsky, yangxiao}@cs.ua.edu

Abstract—Scheduling strategies for on-demand data in broadcast systems typically consider how to minimize the wait time of the requests. When users’ requests for data in a broadcast system have real-time constraints, scheduling strategies for such requests typically only consider how to minimize the number of deadlines missed. There are many applications with both real-time and non-real-time requests that would benefit from a broadcast scheduling strategy that considers both the timing constraints and the wait times of requests. We refer to such a broadcast environment as a mixed-type request broadcast environment. In this paper, we present an on-demand broadcast cost model for mixed-type broadcast environments that considers both the response time and number of deadlines missed. We propose a scheduling strategy for mixed-type broadcast systems, called the maximum paid cost first (MPCF) that is based on this cost model. The simulation results show that our MPCF strategy always achieves the best result for varying request arrival rates, ratio of non-real-time requests and real-time requests, and a weighted missed deadline value, when compared to existing broadcast strategies.

I. INTRODUCTION

In a distributed system, broadcast delivery can be used to efficiently satisfy the requests for data stored at a data server. Broadcasting is advantageous because more than one request can be satisfied by the broadcast of a single data item. Broadcasting has been recognized as particularly beneficial in a wireless environment, where the downlink bandwidth can be much higher than the uplink bandwidth, and broadcasting can help to save the wireless bandwidth and the mobile client’s battery energy. In an on-demand (or pull-based) broadcast approach [1, 3, 4], users send specific data requests to a server and the server only broadcasts the requested data. An algorithm is needed to determine which data to broadcast next, as requests for data are placed in a queue. While there is the cost of an uplink channel for users to send requests, broadcast bandwidth is saved by avoiding broadcasting data that is not needed. Importantly, the on-demand approach can adapt to dynamic changes of data requests when the data access pattern is not static.

In a typical broadcast system, one goal is to minimize the average wait time, which is also referred to as the mean data access time. The data access time (or response time) is the amount of time it takes from the arrival of a page request until the request is satisfied by the broadcast of that page. Many strategies for minimizing the wait time in an on-demand broadcast environment have resulted from research in this area [13-19].

There are also many situations in which a user request for data has a real-time constraint, which is in the form of a deadline. In such systems, minimizing the number of deadlines missed is a goal and the wait time is no longer the focus. There has been some recent work to consider minimizing deadlines missed in an on-demand broadcast environment, [11, 12].

There is a large range of applications that has both real-time and non-real-time requests. We refer to such a broadcast environment with both real-time and non-real-time requests as a mixed-type request broadcast environment. Such a system could benefit from a broadcast system that minimizes both wait time and missed deadlines. In this paper, we propose an on-demand scheduling strategy for mixed-type request broadcast systems. We consider an on-demand strategy in our system since we believe on-demand algorithms better adapt to the dynamic changes in the intensity and distribution of system workloads. We also note that the broadcast strategies we discuss in this research are not restricted in their applications to wireless networks, but are applicable in many different distributed environments, including clients requesting data on a wired LAN.

The remainder of this paper is organized as follows. We present the related work in Section II. In Section III, we provide motivation for this research and problem definition. We propose a cost model named Cost of Balancing the Missing Rate and Response Time and a Maximum Paid Cost First strategy in Section IV. Performance results are shown in Section V. Our conclusions appear in Section VI.

II. RELATED WORK

Scheduling strategies to minimize the wait time for on-demand broadcast systems with non-real-time requests appear in [4, 7, 8]. One of the most popular strategies to address this goal, due to its excellent performance, is Longest Wait First (LWF) algorithm. For the LWF [3, 20], the sum of the total time that all pending requests have been waiting for a data item is calculated, and the data item with the largest total wait time is chosen to broadcast next. LWF makes scheduling decisions based on the current state of a queue and has been shown to outperform all other strategies at minimizing wait time [20].
with the assumption of the page request probabilities following Zipf’s Law [2]. However, LWF has been recognized as expensive to implement. In [4], a strategy called Requests times Wait (RxW) is presented that provides an estimate of the LWF algorithm by multiplying the number of pending requests for a data item times the longest request wait time. In general, the performance of the approximate algorithms has been shown to be close to LWF.

Scheduling data requests with timing constraints in a non-broadcast environment is studied extensively in [5]. A real-time task has timing constraints that are typically in the form of deadlines. There are three different types of deadlines: soft, hard, and firm [22]. If a soft deadline is missed, the result produced still has some value, although the value monotonically decreases with time after the deadline. If a hard deadline is missed, the consequence can be catastrophic, so that the value assigned to a result when a deadline is missed is negative. For a firm deadline, the consequences of missing a deadline are less severe than a hard deadline, but a result produced after a deadline is useless, and is assigned a value of zero. For both firm and hard deadlines, if a deadline is not met, the task is discarded as soon as the deadline is missed, while soft tasks are typically allowed to run until they finish.

In addition to timing constraints, a real-time system, such as a real-time database, can also have temporal consistency constraints requiring temporal data to be up-to-date [21]. The temporal data in such a system can be sensor data, which is derived from sensors monitoring the real-world, or derived data, which can be derived from sensor or other derived data. Temporal data can have a timestamp, which indicates the time the data value was observed in the real-world, and it can also have a validity interval, indicating the time frame during which the data value is valid.

Two of the most popular real-time scheduling strategies to minimize missed deadlines are Earliest Deadline First (EDF) and Least Slack (LS) first [7]. As its name implies, for the EDF, the task with the earliest deadline is given the highest priority. For the LS, the slack time is defined as: \( d - (t + E - P) \), where \( d \) is the deadline, \( t \) is the current time, \( E \) is the execution time and \( P \) is the processor time used thus far. If the slack time is \( \geq 0 \), it means that the task can meet its deadline if it executes without interference. The slack time indicates how long a task can be delayed and still meets its deadline. The LS differs from the EDF because the priority of a task depends on the service time which it has received. If a task is restarted, its priority will be changed. Simulation results show that the EDF is the best overall policy for real-time database systems in a non-mobile environment. However, when system loads are high, the LS and EDF strategies lose their advantages, even over FCFS, as most transactions are likely to miss their deadlines.

Recently, there has been some work addressing on-demand broadcast with timing constraints. In [11] an on-demand broadcast strategy to minimize the percentage of missed deadlines, called Aggregated Critical Requests (ACR) is presented. This heuristic approach in [11] is based on a Markov Decision Process (MDP) model that maximizes the reward by broadcasting the page that will miss the most deadlines if it is not broadcast in the next time slot. Performance results demonstrate that ACR performs better than existing real-time scheduling strategies.

III. MOTIVATION AND PROBLEM DEFINITION

A. Motivation

As we mentioned in previous sections, research has been done on either improving the system response time or reducing the deadline missed rate for data broadcast using on-demand strategies. When improving the system response time, the assumption is that all the requests have no time constraints, while for improving the deadline missed rate, the assumption is that all of the requests have time constraints. In this paper, we consider a more comprehensive broadcast environment in which data requests are of mixed types. There are many reasons to consider the broadcast of mixed-type requests as follows:

First, from a traffic flow perspective, requests for the same data object from different traffic flows may have different requirements. For example, data requests for exit information about an interstate in a transportation system from drivers currently on the interstate are more likely to have a time constraint. On the other hand, requests for the same information from a person who is planning a trip in a nearby café may have no time constraints.

Secondly, from a data object perspective, the requests for temporal data objects and non-temporal data objects may result in different request requirements. For instance, a query about mile markers of the exits on an interstate may not be a real-time request, but a request for the current travel time from Exit A to Exit B can be a real-time request because the current travel time will change with time.

Thirdly, a combination of the traffic flow and data object perspectives is more likely in practice. For example, the above traffic information can be temporal data that changes with time, and the clients who need such information will also have different timing requirements associated with their requests.

Due to the above reasons, we believe that it is important to address mixed-type traffic, and this is the motivation of this work.

B. Problem Definition

As illustrated above, in many cases the request types can be mixed. For the requests without time constraint, the goal is to minimize the amount of time it takes to satisfy the request (Response Time), while for the requests which have a specified time constraint, the goal is to minimize the missed deadlines (Request Drop Rate). There is a tradeoff between the response time and the request drop rate. For the mixed-type broadcast environment, these are two metrics for measures of system performance. It is very difficult, if possible, to minimize the response time and the request drop rate at the same time because these goals conflict with each other.

We assume that the broadcast time is divided into time slots and only one data object is allowed to be broadcast during each time slot. A request which has a time constraint is called a real-time (R) request and a request which has no time constraint
is a non-real-time (NR) request.

Let \( w_{NR}(t,i) \) denote the number of NR requests for data object \( i \) observed at time slot \( t \). Let \( d_{R}(t,j) \) denote the minimum number of real-time requests which will miss their deadlines if page \( j \) is not served at time slot \( t \). Let \( p_{NR}(t) \) denote data object \( i \) which has the maximum \( w_{NR}(t,i) \) observed at time slot \( t \). Let \( p_{R}(t) \) denote the data object \( j \) which has the maximum \( d_{R}(t,j) \) in the time slot \( t \).

We define a static request set as the case where we assume that a request queue is static, such that there will be no new requests arriving.

**Remark 1:** In a mixed type broadcast environment with a static request set, if there is a scheduler which can minimize the response time of NR requests and minimize the request drop rate of R requests at the same time, the scheduler should satisfy: 1) always broadcast \( p_{NR}(t) \) and \( p_{R}(t) \) during time slot \( t \), and 2) \( p_{NR}(t) = p_{R}(t) \) during the entire broadcast time. The above remark is in fact a greedy approach, i.e., it is a local optimization instead of a global optimization. We explain the reason behind the remark as follows. If the broadcast scheduler violates condition 1), there is a time slot \( t \), in which the schedule does not broadcast \( p_{NR}(t) \) or \( p_{R}(t) \), so that it must broadcast another data object, denoted as \( k \), which has \( w_{NR}(t,k) < \max[w_{NR}(t,i)] \). As a result, the total waiting time will be increased because there are more NR requests waiting for at least one more time slot to be broadcast. If it does not broadcast \( p_{R}(t) \) during time slot \( t \), a data object \( k \) has been broadcast whose \( d_{R}(t,k) < \max[d_{R}(t,j)] \), and there will be at least one more R request than in the optimal schedule which will miss its deadline. Condition 2) is the same as condition 1) because we assume only one page will be broadcast during one time slot.

Our goal is to minimize response time and request drop rate by exploring the tradeoff between them. In the next section we present a cost model to balance this tradeoff.

IV. COST MODEL AND MAXIMUM PAID COST FIRST STRATEGY

A. Cost Model

In the above, we can see the schedule must meet strict requirements in order to obtain the optimal scheduling with the assumption there is a static request set. However, realistically, requests can arrive at any time, so the scheduler cannot determine \( d_{R}(t,j) \) because the scheduler lacks future knowledge. This is one reason that it is difficult, if possible, to obtain a global optimization. To balance trading off the response time and request drop rate, we introduce a cost model named Cost of Balancing the Missing Rate and Response Time (CBMR). To the best of our knowledge, we are the first to introduce a cost model for a mixed-type broadcast. It is defined as follows.

For an R request, if it misses its deadline, it will contribute cost \( \alpha \). In our model we assume that all deadlines are firm deadlines and are deleted from the system after a deadline is missed.

For a request \( i \), if the request is an NR request, we denote it as \( i \in NR \); otherwise, we denote it as \( i \in R \). For any request \( i \), let \( a_{i} \) denote the request arrival time. For any R request \( i \), let \( d_{i} \) denote the deadline time, where \( d_{i} > a_{i} \).

For an NR request \( i \), let \( w_{i} = t - a_{i} \) denote the waiting time, where \( t \) is the current time, and if the waiting time is over some waiting threshold \( T_{h} \), it is associated a potential cost factor \( \mu \). When the page is served, it will contribute the cost \( w_{i}(t) - T_{h} \mu \).

Let \( N(t) \) denote the total number of pending requests (both R and NR) at time \( t \). Let \( \cos t(i,t) \) denote the cost for request \( i \) at time \( t \). Let \( C_{BMRC}(t) \) denote CBMR at time \( t \). We have

\[
\cos t(i,t) = \begin{cases} 
\alpha \psi, & i \in R, \{i \in R \} \wedge \{T_{Slack}(i,t) \geq 0\} \\
\alpha, & i \in R, \{i \in R \} \wedge \{T_{Slack}(i,t) < 0\} \\
0, & i \in NR \wedge \{w_{i}(t) > T_{h}\} \\
[w_{i}(t) - T_{h}]\mu, & i \in NR \wedge \{w_{i}(t) \leq T_{h}\} 
\end{cases}
\]

\[
T_{Slack}(i,t) = d_{i} - t
\]

\[
w_{i}(t) = t - a_{i}
\]

\[
C_{BMRC}(t) = \sum_{i=1}^{N(t)} \cos t(i,t)
\]

where \( \psi \) is a discount parameter, such that the cost of a request is higher (lower) if it has a shorter (longer) slack time. \( T_{Slack}(i,t) \) is request \( i \) ’s slack time at time \( t \), which is the longest time the request can stay in the queue without missing the deadline. In equation (1), when \( T_{Slack} < 0 \), we set the cost\( \cos t(i,t) \) to \( \alpha \). This means that for all the real-time requests whose deadlines are missed, they will be associated with a cost of \( \alpha \) at any time after their deadline. However, for firm deadlines (as assumed in this paper), once a real-time request misses its deadline, it will be removed from the broadcast request queue and the cost of \( \alpha \) for \( \{i \in R \} \wedge \{T_{Slack}(i,t) < 0\} \) will not occur.

Typically, the drop rate for an R request will be more important than the waiting time for an NR request. Hence, we will introduce the parameters \( K \) and \( C \) to represent the relationship between \( \alpha \) and \( \mu \).

\[
\alpha = K\mu + C, \text{ where } \mu \geq 0, K > 0, C \geq 0.
\]

We now use \( C_{BMRC}(t) \) to measure the system performance in a mixed-type broadcast environment.

B. Maximum Paid Cost First Strategy

In the previous section, we have introduced a new performance metric in the mixed type broadcast environment. This cost model is more general for a broadcast environment because it can be applied to the different types of broadcast environment we discussed (non-real-time, real-time or mixed). The scheduling problem of obtaining an optimal solution is a typical NP-hard problem; even when the request set is static. For online scheduling without future knowledge, it is difficult to implement a scheduling strategy to achieve the optimal results. We now propose a heuristic strategy to schedule the broadcast of mixed-type requests. This strategy is called the Maximum Paid Cost First (MPCF) as follows.

To minimize the CBMR, we can model the problem as a priority queue in which the broadcast scheduler will broadcast
the page with the highest priority each time. We now discuss how to set the priority of each request queue.

We need to set the priority of each page dynamically, in order to consider both the request drop rate of the real-time requests and the waiting time for the non-real-time requests.

Let \( S(j, t) \) denote the set of requests (both R and NR) at time \( t \) in the waiting queue for page \( j \). Assume that the time to broadcast a page is \( \Delta t \) time. The proposed scheme will broadcast page \( k \), where

\[
k = \max_j \left\{ \sum_{i \in S(j, t)} \cos (i, t + \Delta t) \right\}
\]

(6)

The proposed strategy serves the page with the maximum cost.

V. PERFORMANCE EVALUATION

A. Simulation Setting

In this study we compare our MPCF approach to the FCFS, LWF and RxW algorithms:

- **FirstComeFirstServed (FCFS):** Data items are broadcast in the order of their requests.
- **Longest Wait First (LWF):** selects the page that has the longest total waiting time.
- **Request Count Times Waiting Time (RxW):** selects the page that has the greatest value for: number of requests in queue times longest request waiting time in queue.

We do not include other scheduling algorithms, such as EDF, SIN and ACR-B in our comparison. They will not fit into our mixed-type request environment because all three of these scheduling algorithms can only weight the requests that have time constraints. These three pure real-time scheduling algorithms cause non-real-time requests to starve if they are applied in the mixed type request environment.

We assume a broadcast architecture that contains the necessary components for an on-demand broadcast system. All data items are stored in a data server in a fixed location. Clients send requests to the server via an uplink channel before the requested page can be broadcast. Newly generated data requests are sent to the server immediately, and the request-generating time is equal to the time the server receives the request (network delay is ignored). We also ignore the overhead of request processing at the server, because the main purpose of the model is to compare the scheduling power of various strategies. We assume all data items have equal size and, likewise, equal service time.

In order to evaluate our MPCF strategy, similar to [12], we utilize real traces collected from the World Cup '98 website [9] to simulate requests made by clients. We illustrate the experimental results using the trace of day-38. As shown in Table 1, it contains more than 7 million requests for 4923 distinct web pages, with an average request rate of 83 requests per second. The access pattern follows a Zipf-like distribution [2]. The arrival rate of requests ranges from 20 to 1280. The number of pages that are transmitted per second is assumed to be 10 pages per second. Deadlines for requests are assigned using the distributions of exponential, uniform or fixed. Similar to [12], we assume a mean relative deadline of 60. The exponential distribution assumes rate \( \lambda = 1/60 \); the uniform distribution assumes deadlines randomly distributed between 2 and 120; and the fixed distribution assumes a deadline of 60.

We assume all R requests have a firm deadline. We list the simulation parameters in Table 1.

<p>| TABLE 1: DEFAULT SIMULATION SETTINGS: |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Request Arrival Rate</td>
<td>80/ time slot</td>
</tr>
<tr>
<td>Average Broadcast Rate</td>
<td>10 data object/time slot</td>
</tr>
<tr>
<td>Average Deadline</td>
<td>60 time slots</td>
</tr>
<tr>
<td>Total Number of Data Objects</td>
<td>4923</td>
</tr>
<tr>
<td>Total processed requests</td>
<td>300,000</td>
</tr>
<tr>
<td>Ratio of Number Real time request and non real time request</td>
<td>50%</td>
</tr>
<tr>
<td>( T_\alpha )</td>
<td>60 time slots</td>
</tr>
<tr>
<td>( w )</td>
<td>0.7</td>
</tr>
<tr>
<td>( K )</td>
<td>200</td>
</tr>
<tr>
<td>( \mu )</td>
<td>1</td>
</tr>
<tr>
<td>( C )</td>
<td>0</td>
</tr>
<tr>
<td>( \Delta t )</td>
<td>1 time slot</td>
</tr>
</tbody>
</table>

B. Impact of varying arrival rate

We compare the system performance of the MPCF, FCFS, LWF [20] and RxW [4] scheduling strategies in a mixed-type broadcast environment by measuring the cost with request arrival rates ranging from 40-640 requests per time slot. The results are illustrated in Figs.1-3. As shown in these figures, our MPCF approach always achieves the lowest cost, no matter the type of deadline distribution. The FCFS always performs the worst. The highest costs overall occur for exponentially distributed deadlines and the lowest costs for fixed deadlines.

Our MPCF will gain at least 20% over the second best algorithms for any arrival rate and deadline distribution. With an increase in arrival rate, the cost will increase. This is obvious since the system can only serve one page at any time slot. A higher arrival rate means more non-real-time requests will remain in the queue and more real-time requests will miss their deadlines, which in turn will bring a higher cost.
C. Impact of varying the real-time to non-real-time ratio

We now discuss how the ratio of the number of real-time requests to the number of non-real-time requests will impact the performance algorithms. Figs. 4, 5 and 6 demonstrate the results for the MPCF, FCFS, RxW and LWF strategies for the three different deadline distributions. The value of the ratio ranges from 0.1 to 0.9.

As indicated by the results, our MPCF algorithm will always perform the best of the four algorithms, regardless of the ratio and the deadline distribution. We also note that the greater the ratio of the real-time requests to non-real-time requests, the higher the cost will be, except for FCFS with a fixed deadline. FCFS will have almost the same cost level as the ratio increases because with a higher ratio, a real-time request is more likely to be chosen. The simulation shows that the total waiting time of non-real-time requests drastically increases.

D. Impact of varying the K-value

In previous sections, we mentioned that the cost paid for missing one real-time request will typically be much more than letting a non-real-time request wait one more time slot in the queue. Now, we will evaluate how the K-value in equation (5) will affect the performance of the scheduling algorithms. We illustrate the simulation results in Figs. 7, 8 and 9 as the values of K are varied from 50 to 600.

All of the other three scheduling algorithms, except for MPCF, will have a higher cost as K increases. This is because the larger K is, the higher the average cost needed to pay for the missed real-time requests. Our MPCF algorithm still achieves the best results for all the deadline distributions.
VI. CONCLUSION

In this paper, we introduced a mixed-type request broadcast environment. We specified the two conditions a broadcast scheduler should meet in order to achieve the minimum request drop rate and the response time in a mixed type request environment with the assumption of no new arrival requests. We described the tradeoff between the request drop rate of the real-time requests and the response time of the non-real-time requests. We then introduced a novel cost model for a mixed-type request environment and also proposed a new scheduling strategy to minimize the cost of broadcast scheduling based on this cost model. The simulation results show that our strategy always achieves the best result with varying request arrival rates, ratio of non-real-time requests and real time requests, and the K-value. This performance improvement over other strategies results from our algorithm considering both the cost of missed deadlines as well as the cost of waiting for a response.

In the future, we will continue work on the system model by utilizing Markov decision process (MDP), and design additional scheduling strategies to achieve a lower cost based on our cost model. The computation overhead at each broadcast slot will be addressed and some improvements will be proposed. In addition, more simulations will be conducted based on different data sets.

REFERENCES