On Scheduling Time-Critical On-Demand Broadcast

Jianliang Xu, Xueyan Tang, Wang-Chien Lee

***Abstract*— On-demand broadcast is an effective data dissemination technique to enhance system scalability and deal with dynamic user access patterns. With the rapid growth of time-critical information services and emerging applications such as mobile location-based services, there is an increasing need for the system to support timely data dissemination. This paper studies online scheduling algorithms for time-critical on-demand broadcast. We propose a novel scheduling algorithm called SIN-α that takes into account the urgency and productivity of serving pending requests. An efficient implementation of SIN-α is presented. Moreover, we analyze the optimal broadcast schedule in terms of request drop rate when the request arrival rate rises towards infinity. Trace-driven experiments demonstrate that SIN-α significantly outperforms existing algorithms over a wide range of workloads. The results also show that the performance of SIN-α approaches the analytical optimum at high request rates.**

***Index Terms*— Mathematical optimization, simulation.**

I. Introduction

The ever growing popularity of the Internet and the resultant slow responses perceived by users have given rise to vast research efforts on improving the performance of web accesses. As the system scale and user base continue to grow, there is an increasing demand for information providers to be capable of concurrently delivering a large amount of information to a huge number of users, especially in popular events such as elections and Olympics games. As a result, innovative delivery technologies, including satellite communications (e.g., StarBand [22] and DIRECWAY [23]), cable networks, and wireless networks (e.g., 2.5G and 3G), have been developed and deployed to provide shared broadband Internet accesses.

Different from traditional networks, a distinguished feature of these new technologies is that they naturally support *broadcast*. In contrast to *unicast* where a data item of interest to multiple clients must be sent individually to each client, broadcast satisfies *all* outstanding requests for the same object by a single transmission, leading to more efficient use of shared bandwidth [1]. In general, there are two broadcast approaches: *push-based broadcast* computes the broadcast program based on historical statistics; *on-demand broadcast* schedules broadcast items based on outstanding requests. While push-based broadcast is useful for certain applications (e.g., small databases with stable access patterns), on-demand broadcast is more widely used for dynamic, large-scale data dissemination like that in the Internet.

With the rapid growth of time-critical information services and business-oriented applications, there is an increasing demand to support quality of service (QoS) in content distribution [20]. In many situations, user requests are associated with time constraints as a measure of QoS. These constraints can be imposed either by the users or the applications. Consider a driver who queries a traffic information server to select one of several alternative routes at some point ahead [13]. Clearly, it is necessary for the server to provide the driver with the traffic information (e.g., which route is less congested) *before* he reaches that point; otherwise, the information is of no value to the driver. As another example, HTTP requests are normally associated with timeout values to prevent users from unlimited waiting when the web servers are heavily loaded. Furthermore, in mobile location-based information services [17], the query response (e.g., the nearest bus station) is useful to a mobile user only if it arrives soon after the query issuance; otherwise, the user may have moved away from the original location such that the response is invalid with respect to the user's new location. In all the above cases, it is necessary for users to specify for each request a *deadline* beyond which she is no longer interested (or less interested) in the requested information. This paper focuses on on-demand broadcast with time constraints, which we shall refer to as *time-critical on-demand broadcast.*

A key issue in the design of an on-demand broadcast system is the *scheduling algorithm* used to select and broadcast requested items from outstanding requests. While there has been significant work on the development of on-demand broadcast scheduling algorithms (e.g., [2], [3], [24]), none of them considered the time constraints associated with requests. On the other hand,

Jianliang Xu is with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong. Email: xujl@comp.hkbu.edu.hk. Xueyan Tang is with the School of Computer Engineering, Nanyang Technological University, Singapore. Email: asxytang@ntu.edu.sg. Wang-Chien Lee is with the Department of Computer Science and Engineering, Penn State University, University Park, PA. Email: wlee@cse.psu.edu.

although time-critical scheduling algorithms have been investigated for unicast-based real-time systems and push-based broadcast systems (e.g., [5], [11], [16]), they are not applicable or not effective to on-demand broadcast systems. In this paper, we are interested in developing new scheduling algorithms for time-critical on-demand broadcast.

The main contributions of this paper are three-fold:

- This is the first effort, to the best of our knowledge, to take into account the time constraints in on-demand broadcast scheduling algorithms. We propose a low-complexity scheduling algorithm, called *SIN-*α, for time-critical on-demand broadcast.
- We analyze the optimal broadcast schedule in terms of request drop rate when the request arrival rate rises towards infinity. The analytical results can be used as a yardstick in experimental evaluations. It can also be employed to facilitate bandwidth allocation in planning and provisioning of on-demand broadcast.
- We conduct trace-driven simulation experiments to study the performance of the proposed scheduling algorithm under a wide range of workloads and show that the proposed SIN-α algorithm significantly outperforms existing algorithms and, at high request rates, approaches the analytical optimum.

The rest of this paper is organized as follows. Section II summarizes the related work. The system model is described in Section III. Section IV presents the proposed scheduling algorithm, SIN-α, together with the implementation issues. Section V analyzes the optimal broadcast schedule when the request rate rises towards infinity. Section VI experimentally compares the performance of the proposed algorithm with the existing algorithms and the analytical optimum. Finally, Section VII concludes the paper.

II. Related Work

There has been significant work on the development of on-demand broadcast scheduling algorithms [24]. Recently, Acharya and Muthukrishnan [2] studied some scheduling algorithms for variable-size data items and introduced a new performance metric called *stretch*. Aksoy and Franklin [3] proposed a low overhead and scalable scheduling algorithm called R$\times$W. Datta *et al.* [10] took into consideration the energy saving issue in the design of on-demand broadcast systems. Liberatore [18] studied the scheduling algorithms for requests asking for a list of dependent items. Hu and Chen [14] investigated dynamic traffic-aware scheduling algorithms. However, none of these algorithms considered the time constraints of requests in making scheduling decisions.

There exist a few studies on broadcast scheduling with time constraints [6], [16]. However, these studies considered periodic push-based broadcast only, which is fundamentally different from on-demand broadcast in system architecture. Xuan *et al.* [26] evaluated several alternative system designs for time-constrained data accesses and showed that on-demand broadcast with the earliest deadline first (EDF) scheduling algorithm performs well. Fernandez and Ramamritham [13] studied an *adaptive* hybrid broadcast system in which the EDF algorithm is employed for on-demand broadcast. These two studies concentrated on the system design aspect of on-demand broadcast, which is complementary to the focus of this paper on the algorithmic aspect of time-critical on-demand broadcast.

Other closely related areas include task scheduling in real-time systems and transaction processing in real-time databases. Many algorithms and theoretical results have been developed [8], [11]. One of the most classical scheduling algorithms is the EDF algorithm [19]. It offers the optimal performance in light-load conditions in terms of deadline missing rate. In overloaded conditions, not all tasks can be completed by their deadlines. When task service times are not available, EDF performs poorly because it may favor the tasks whose remaining lifetimes are shorter than their service times. As a result, these tasks would not only miss their deadlines but also waste system resources to prevent more tasks from meeting their deadlines. Various techniques have been proposed to handle this "domino effect" [11]. Unfortunately, all these existing scheduling algorithms are designed for a *unicast* environment where a newly arrived task cannot be combined with any existing tasks. In contrast, our paper focuses on a broadcast environment where a new request can be merged with an existing outstanding request if the same item is requested. Both requests would be satisfied by a single broadcast of the item. This fundamental difference in system model motivates us to develop new scheduling algorithms.

Closest to our scheduling problem is the value-deadline task scheduling in real-time systems. In this approach, each task is characterized by an importance value and the scheduling algorithm aims to maximize the cumulative value of the tasks that meet their deadlines [8]. Our problem resembles value-deadline scheduling in that the number of requests for a data item can be treated as the value of the item. However, in an on-demand broadcast system, the number of requests for an item may change over time due to new arrivals and deadline expiration. This is different from value-deadline scheduling where the values are constants. As shall be shown in Section VI, a good value-deadline scheduling

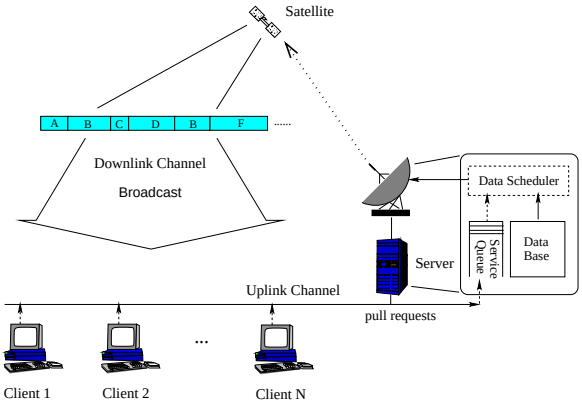


Fig. 1. A Satellite-based Broadcast Architecture

algorithm (i.e, *value-density scheduling* [8], which is analogous to the most requests first (MRF) algorithm for fixed-size items) shows poor performance for on-demand broadcast scheduling.

Other related work concerning on-demand broadcast includes energy-efficient retrieval [9], [25], client cache management [21], combination with pushed broadcast [15], and fault-tolerant broadcast [5]. These studies complement to our work in different aspects.

In summary, existing time-critical scheduling algorithms are confined to push-based broadcast and unicast systems only. Meanwhile, existing on-demand broadcast scheduling algorithms ignore the time constraints associated with requests. This paper, to the best of our knowledge, investigates the scheduling algorithms for time-critical on-demand broadcast for the first time in the literature.

III. System Model

As shown in Figure 1, we consider a satellite-based broadcast architecture that captures all essential components of a typical on-demand broadcast system [3]. In this architecture, a large group of clients retrieve data items (e.g., traffic information) maintained by a data server. The clients send requests to the server through an uplink channel. Each request is characterized by a 3-tuple: $< id, t, d >$, where id is the identifier of the requested item, t is the time of request, and d is a *relative deadline*. The *absolute (service) deadline* of a request is given by $t + d$, beyond which the receipt of the requested item is of no value to the client. The client monitors a downlink broadcast channel for the requested item until the item is broadcast or the lifetime of the request expires. The uplink and downlink channels are independent.

On receiving a request, the server inserts it into a service queue. An outstanding request is said to be *active* if its lifetime has not expired. An active request is called *feasible* if it is still possible to meet its service deadline. Otherwise, if its service deadline cannot be satisfied, the active request is *degenerated*. Active requests remain in the service queue until they are serviced or degenerated, whichever takes place earlier.

All data items are assumed to be locally available on the server. The server broadcasts data items chosen from feasible requests based on a scheduling algorithm. The primary goal of a scheduling algorithm is to satisfy as many requests as possible. This is best measured by *request drop rate*, which is defined as the ratio of the number of requests missing their deadlines to the total number of requests. The selected items are sent to the network controller for broadcast and the associated requests are removed from the service queue. In this paper, we focus on new factors that affect the performance of time-critical broadcast scheduling in addition to those previously considered in unicast scheduling. For simplicity, all data items are assumed to have equal size (and hence equal service time). Note that the factor of item size (or service time) can be easily incorporated in broadcast scheduling algorithms to handle variable-size data items [2].

IV. Proposed SIN-α Algorithm

In this section, we propose a new scheduling algorithm with low-complexity, called SIN-α. We start by illustrating the factors affecting the performance of time-critical broadcast scheduling. The broadcast duration of an item is referred to as a *broadcast tick*. We compare EDF (earliest deadline first) and MRF (most requests first), two typical scheduling algorithms in unicast and broadcast respectively. At each broadcast tick, EDF broadcasts the item with the shortest remaining lifetime to cater for the *urgency* of requests. MRF, on the other hand, broadcasts the item that has the largest number of pending requests to account for the *productivity* of broadcasting. As shall be shown in Section VI, EDF and MRF respectively achieve good performance for certain workloads only. This motivates us to integrate the *urgency* and *productivity* factors to improve scheduling performance. Intuitively,

- For two items with the same number of pending requests, the one with closer deadline should be broadcast first.
- For two items with the same deadline, the one with more pending requests should be broadcast first.

Motivated by the above observations, we propose a new scheduling algorithm, called SIN-α (*S*lack time

Inverse Number of pending requests).[1] Specifically, the $sin.\alpha$ value of each item that has at least one pending request is given by

$$sin.\alpha = \frac{slack}{num^{\alpha}} = \frac{1stDeadline - clock}{num^{\alpha}},$$

where *slack* is the duration from the current time (i.e., $clock$) to the deadline of the most urgent pending request for the item (i.e., $1stDeadline$), num is the number of pending requests for the item, and $\alpha \geq 0$ is a relative weight of productivity to urgency. At each broadcast tick, the item with the minimum $sin.\alpha$ value is broadcast on the downlink channel. It is easy to see that the larger the value of α, the more influential the number of pending requests.

We remark here that, in SIN-α, the earliest deadline of pending requests rather than the mean/median deadline is used as an estimate of urgency. This is because in the context of time-critical broadcast, the earliest deadline reflects the urgency of satisfying *all* requests for the item, but the mean/median deadline reflects that of satisfying only the requests whose deadlines are beyond the mean/median deadline. In preliminary experiments, we have observed that the SIN-α algorithm with the earliest deadline performs better than that with the mean/median deadline.

A. An Efficient Implementation

A straightforward implementation of SIN-α is to compute, at each broadcast tick, the $sin.\alpha$ values of all items that have pending requests and to broadcast the one with the minimum $sin.\alpha$ value. Such an implementation has a scheduling complexity of at least $\mathcal{O}(m)$, where m is the number of items with pending requests. In the following, we present a more efficient implementation of SIN-α, which was inspired by [3].

We group the pending requests in the service queue by the requested items. Two data structures, an *S-list* and an *N-list*, are used to index the requested items in the service queue. Each item has one entry in the S-list and N-list respectively. As shown in Figure 2, the S-list is a bidirectional linked list where the items are sorted in ascending order of the associated earliest deadline (i.e., in ascending order of $slack$). In the N-list, the items having the same number of pending requests are first structured into a min-heap built on the key of the earliest deadline. The roots of the heaps are then

[1]Note that there are other ways to combine the factors of slack time and number of pending requests in scheduling. We advocate the SIN-α algorithm because of its simplicity, efficient implementation (see the next subsection), and demonstrated excellent performance (see Section VI).

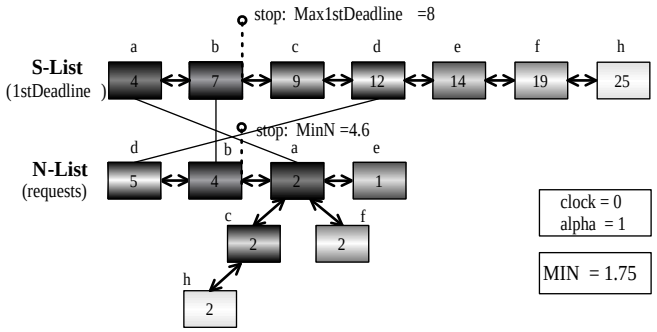


Fig. 2. Indexing Structures of the Service Queue

organized into a bidirectional linked list in descending order of the number of pending requests (i.e., num). The heap that indexes the items with n pending requests is referred to as *heap-n*.

At each broadcast tick, the server broadcasts the item with the minimum $sin.\alpha$ value. The proposed data structures reduce the search space of candidate items in two aspects. First, since the requested items in each min-heap of the N-list have the same number of requests and the min-heap is constructed based on the key of the earliest deadline, the root item of each heap has the minimum $sin.\alpha$ value among all the items in the heap. Thus, non-root items in the N-list can be excluded from the search space. Moreover, the search space can be further pruned by searching the S-list and N-list in an alternate fashion. Starting from the heads of the two lists, we sequentially examine the entries therein. Two values, $MinN$ and $Max1stDeadline$, are maintained to cut off the search space in the N-list and S-list respectively. Let MIN denote the minimum $sin.\alpha$ value found so far. Since the S-list is sorted in ascending order of $slack$, an unexamined item has a $sin.\alpha$ value less than MIN only if its num value exceeds

$$MinN = \left(\frac{NextS}{MIN}\right)^{\frac{1}{\alpha}},$$

where $NextS$ is the $slack$ value of the next item in the S-list. Similarly, since the N-list is sorted in descending order of num, an unexamined item has a smaller $sin.\alpha$ value than MIN only if its $slack$ value is less than

$$MaxS = (NextN)^{\alpha} \cdot MIN,$$

i.e., the corresponding $1stDeadline$ value is less than

$$Max1stDeadline = (NextN)^{\alpha} \cdot MIN + clock,$$

where $NextN$ is the num value of the next item in the N-list. MIN, $MinN$, and $Max1stDeadline$ are updated after examining each item. The search process continues until the list tails are reached or the next items

in the lists violate the necessary conditions indicated by $MinN$ and $Max1stDeadline$. The pseudo-code of the scheduling algorithm is presented in Algorithm 1, where pn and ps point to the next items in the N-list and S-list respectively.

Figure 2 shows an example. Suppose that the current clock is 0 and α is set to 1. First, we examine the first item d in the N-list and get $MIN = 2.4$, $MinN = 1.7$, and $Max1stDeadline = 9.6$. Then, we go ahead to examine the first item a in the S-list and obtain a smaller $sin.\alpha$ value 2.0 for a. Therefore, we update $MIN = 2.0$, $MinN = 3.5$, and $Max1stDeadline = 8.0$. Next, we go to the second item b in the N-list, whose $sin.\alpha$ value is 1.75, and we update $MIN = 1.75$ and $MinN = 8.0$ ($Max1stDeadline$ remains at 8.0 since pn becomes nil). The searching ceases here since the remaining items do not have a $1stDeadline$ value greater than $Max1stDeadline$ and an N value less than $MinN$ and, hence, cannot have a $sin.\alpha$ value less than MIN. In total, we only need to examine three item entries to find the item b to broadcast.

When a new request arrives, the request is inserted into the service queue and the corresponding request group is updated. If the request group is empty, a new item entry is created for the requested item and two index entries are inserted into the S-list and *heap-1* of the N-list respectively. Otherwise, the earliest deadline of the requested item is updated if necessary, and the S-list is adjusted accordingly. Moreover, the entry of the requested item in the N-list is moved from *heap-*x to *heap-*(x+1), if there were x pending requests for the item before the new request arrival. After selecting an item to broadcast, the scheduler removes from the service queue the requests for the item as well as those whose lifetimes will expire in the next broadcast tick (i.e., degenerated requests).

Algorithm 1 Efficient Search Algorithm for SIN-α.

1: *MIN* := ∞
2: *pn* := the head of the N-list
3: *ps* := the head of the S-list
4: **while** (*pn* $\neq$ *nil* or *ps* $\neq$ *nil*) **do**
5: **if** *pn* $\neq$ *nil* **then**
6: calculate *sin.*α_{pn}, the $sin.\alpha$ value of the item pointed by *pn*
7: **if** *sin.*α_{pn} $<$ *MIN* **then** *MIN* := *sin.*α_{pn}
8: **if** *ps* and *pn* refer the same item **then** advance *ps* to the next unexamined item in the S-list whose entry in the N-list is a heap root
9: advance *pn* to the next unexamined item in the N-list
10: **if** *ps* $\neq$ *nil* **then** *MinN* := $(\frac{ps \rightarrow 1stDeadline - clock}{MIN})^{\frac{1}{\alpha}}$
11: **if** *pn* $\neq$ *nil* and *pn*$\rightarrow$*num* $<$ *MinN* **then** *pn* := nil
12: **if** *pn* $\neq$ *nil* **then** *Max1stDeadline* := (*pn*$\rightarrow$*num*)$^{\alpha} \cdot MIN + clock$
13: **if** *ps* $\neq$ *nil* and *ps*$\rightarrow$*1stDeadline* $>$ *Max1stDeadline* **then** *ps* := nil
14: **end if**
15: **if** *ps* $\neq$ *nil* **then**
16: calculate *sin.*α_{ps}, the $sin.\alpha$ value of the item pointed by *ps*
17: **if** *sin.*α_{ps} $<$ *MIN* **then** *MIN* := *sin.*α_{ps}
18: **if** *ps* and *pn* refer the same item **then** advance *pn* to the next unexamined item in the N-list
19: advance *ps* to the next unexamined item in the S-list whose entry in the N-list is a heap root
20: repeat lines 10-13
21: **end if**
22: **end while**

V. Analytical Results

In this section, we analyze the optimal broadcast schedule in terms of request drop rate when the request arrival rate rises towards infinity. The analytical results will be used as a yardstick in our experimental evaluations. The notations used in the analysis are summarized in Table I.

Notation	Description
N	number of retrievable data items
λ	total request rate
p_i	access probability of item i (w.l.o.g., assume $p_1 \geq p_2 \geq \cdots \geq p_N$)
λ_i	request rate of item i, i.e., $\lambda_i = p_i \cdot \lambda$
s_i	inter-broadcast duration of item i
l	service time of a data item
$F(t)$	CDF of relative deadlines
η_i	request drop rate of item i
$\eta_{[1..N]}$	total request drop rate of all items

TABLE I
Notations Used in Analysis

Consider two consecutive broadcast instances of an item i at times 0 and τ (see Figure 3). Observe that a request arriving in an infinitely short interval $[t, t + \Delta t]$ $(0 < t \leq \tau)$ cannot be satisfied if and only if its relative deadline is within $(\tau - t)$. Moreover, when the request rate approaches infinity, the number of requests for item i arriving in any (infinitely) short duration Δt can be approximated by $\lambda_i \Delta t$, where λ_i is the request rate for item i. Therefore, the drop rate of requests for item i

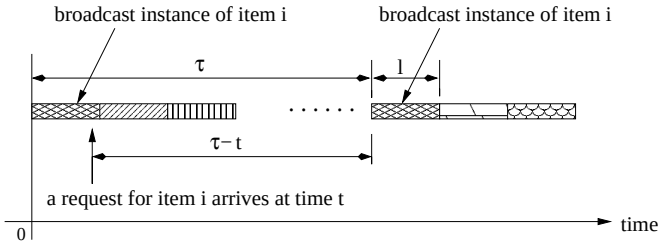


Fig. 3. System Model

arriving in interval $[0, \tau]$ is given by

$$\eta_i = \frac{\int_0^\tau F(\tau - t)\lambda_i dt}{\int_0^\tau \lambda_i dt}, \tag{1}$$

where $F(t)$ is the CDF (cumulative distribution function) of relative deadlines, i.e., the probability that a relative deadline does not exceed t.

We first show that the request drop rate of an item is minimized when the broadcast instances of the item are equally spaced.

Theorem 1: Given a fixed bandwidth share for a data item, periodic broadcast offers the lowest request drop rate.

Proof: See Appendix A. □

Let the inter-broadcast duration of item i be s_i ($i = 1, 2, \cdots, N$). Then, the drop rate of requests for item i is given by

$$\eta_i = \frac{1}{s_i}\int_0^{s_i} F(s_i - t)dt.$$

Therefore, the overall request drop rate is given by

$$\eta_{[1..N]} = \sum_{i=1}^{N}\frac{\lambda_i}{\lambda}\eta_i = \sum_{i=1}^{N}\frac{p_i}{s_i}\int_0^{s_i} F(s_i - t)dt. \tag{2}$$

Let l be the service time of a data item. An inter-broadcast duration of s_i implies a fraction $\frac{l}{s_i}$ of the bandwidth is used to broadcast item i. Therefore,

$$\sum_{i=1}^{N}\frac{l}{s_i} = 1 \qquad \Longrightarrow \qquad \sum_{i=1}^{N}\frac{1}{s_i} = \frac{1}{l}. \tag{3}$$

In the following, we analyze the lowest request drop rate (2) under constraint (3) for three different distributions of relative deadlines: exponential, uniform, and fixed distributions. For simplicity, we assume same deadline distribution for all items.

Exponentially Distributed Relative Deadlines The CDF of an exponential distribution with a mean of M is given by

$$F(t) = 1 - e^{-\frac{t}{M}} \qquad (t \geq 0).$$

Given such a distribution of relative deadlines, the total request drop rate (2) can be re-written as

$$\eta_{[1..N]} = 1 - \sum_{i=1}^{N}\frac{p_i M(1 - e^{-\frac{s_i}{M}})}{s_i}. \tag{4}$$

Theorem 2: Under exponentially distributed relative deadlines with a mean value of M, the total request drop rate is minimized when all inter-broadcast duration s_i's ($1 \leq i \leq N$) satisfy

$$p_i(1 - \frac{s_i}{M}e^{-\frac{s_i}{M}} - e^{-\frac{s_i}{M}}) = K, \tag{5}$$

for some constant K.

Proof: The theorem is proved by applying the Lagrange multiplier method [12]. □

Equation (5) together with constraint (3) can be solved numerically such as using a bisection method.

Uniformly Distributed Relative Deadlines The CDF of a uniform distribution between 0 and L is

$$F(t) = \begin{cases} \frac{t}{L} & 0 \leq t \leq L, \\ 1 & t > L. \end{cases}$$

In this case, the total request drop rate (2) is re-written as

$$\eta_{[1..N]} = \sum_{i=1}^{N} p_i \eta_i,$$

where

$$\eta_i = \begin{cases} \frac{s_i}{2L} & 0 < s_i \leq L, \\ 1 - \frac{L}{2s_i} & s_i > L. \end{cases} \tag{6}$$

Lemma 1: Let $(s_1, s_2, \cdots, s_N)$ be a set of inter-broadcast durations producing the lowest request drop rate. For any two items i and j where $p_i > p_j$, if $s_j \leq L$, it follows that $s_i \leq L$.

Proof: See Appendix B. □

Without loss of generality, we number the items in decreasing order of access probability, i.e., $p_1 \geq p_2 \geq \cdots \geq p_N$. Lemma 1 implies that in the optimal broadcast scheduling, there exists an *identification* item I such that $\forall i \leq I$, $s_i \leq L$, and $\forall i > I$, $s_i > L$.

Lemma 2: There exists a set of inter-broadcast durations $(s_1, s_2, \cdots, s_N)$ producing the lowest request drop rate such that

(i) $\forall i \leq I$, $s_i \leq L$;
(ii) $s_{I+1} > L$;
(iii) $\forall i > I + 1$, $s_i = \infty$ (i.e., $\frac{1}{s_i} = 0$);

where I is the identification item.

Proof: See Appendix C. □

Lemma 3: Let I be the *identification* item. Given a fixed bandwidth share $f \geq \frac{I \cdot l}{L}$ for items $1, 2, \cdots, I$. The optimal set of inter-broadcast durations minimizing the request drop rate of these items

$$\eta_{[1..I]} = \sum_{i=1}^{I} \frac{p_i s_i}{2L} \tag{7}$$

subject to the constraints

$$\sum_{i=1}^{I} \frac{1}{s_i} = \frac{f}{l}$$

and

$$\forall 1 \leq i \leq I, s_i \leq L$$

is given by

$$s_i = \begin{cases} \frac{\sum_{j=1}^{m} \sqrt{p_j}}{(\frac{f}{l} - \frac{I-m}{L})\sqrt{p_i}} & i \leq m, \\ L & m < i \leq I, \end{cases}$$

where m = max$\{x \mid \frac{\sum_{j=1}^{x} \sqrt{p_j}}{(\frac{f}{l} - \frac{I-x}{L})\sqrt{p_x}} \leq L\}$. The lowest drop rate is given by

$$\frac{1}{2L(\frac{f}{l} - \frac{I-m}{L})}\Big(\sum_{i=1}^{m} \sqrt{p_i}\Big)^2 + \sum_{i=m+1}^{I} \frac{p_i}{2}.$$

Proof: See Appendix D. □

Lemmas 1 to 3 imply the lowest request drop rate can be computed in a brute-force fashion. Thus, we have the following theorem:

Theorem 3: Under uniformly distributed relative deadlines between 0 and L, the lowest request drop rate is given by

$$\min_{0 \leq I \leq N, f \geq \frac{Il}{L}, f > 1 - \frac{l}{L}} \Big\{ \frac{1}{2L(\frac{f}{l} - \frac{I-m}{L})}\Big(\sum_{i=1}^{m} \sqrt{p_i}\Big)^2 + \sum_{i=m+1}^{I} \frac{p_i}{2} + p_{I+1}(1 - \frac{L}{2} \cdot \frac{1-f}{l}) + \sum_{i=I+2}^{N} p_i \Big\},$$

where m = max$\{x \mid \frac{\sum_{j=1}^{x} \sqrt{p_j}}{(\frac{f}{l} - \frac{I-x}{L})\sqrt{p_x}} \leq L\}$.

Fixed Relative Deadlines The CDF of fixed relative deadline C is

$$F(t) = \begin{cases} 0 & 0 \leq t \leq C, \\ 1 & t > C. \end{cases}$$

Therefore, the total request drop rate (2) can be re-written as

$$\eta_{[1..N]} = \sum_{i=1}^{N} p_i \eta_i,$$

where

$$\eta_i = \begin{cases} 0 & 0 < s_i \leq C, \\ 1 - \frac{C}{s_i} & s_i > C. \end{cases} \tag{8}$$

Theorem 4: Under fixed relative deadline C, the total request drop rate is minimized when the inter-broadcast duration s_i's satisfy

$$s_i = \begin{cases} C & i \leq \lfloor y \rfloor, \\ \frac{C}{y - \lfloor y \rfloor} & i = \lfloor y \rfloor + 1, \\ \infty & i > \lfloor y \rfloor + 1, \end{cases}$$

where $y = \frac{C}{l}$. The lowest drop rate is given by

$$\eta^* = p_{\lfloor y \rfloor + 1}(1 - y + \lfloor y \rfloor) + \sum_{i=\lfloor y \rfloor + 2}^{N} p_i.$$

Proof: It is easy to infer that the lowest drop rate is achieved by assigning the most popular items an inter-broadcast duration of C subject to the total bandwidth constraint. □

In addition to serving as a yardstick in performance evaluations, the analysis presented in this section can also be used to facilitate bandwidth allocation in planning and provisioning of on-demand broadcast. Specifically, given the access pattern and the deadline distribution of requests, the analytical results can be employed to estimate the bandwidth required to achieve a desired level of request drop rate at high request rates.

VI. Performance Evaluation

A. Experimental Setup

A trace-driven simulator has been developed to evaluate the performance of the proposed algorithm. The simulator models the architecture presented in Section III. Real traces collected from the World Cup '98 website [27] were used to simulate the requests made by clients. Similar performance trends were observed for different daily traces. Due to space limitation, we shall report the experimental results of one trace (i.e., the day-38 trace) only in this paper. The day-38 trace contains over 7 million requests for 4923 distinct web pages.[2] The average request rate is 83 per second. The access counts of different pages sorted in descending order are shown in Figure 4. It can be seen that the access pattern follows a Zipf-like distribution, which is consistent with the observation made in the literature [7]. To simulate different levels of workloads, we changed the time scale of the trace by introducing a *request*

[2]Interested readers are referred to [4] for more details of the WorldCup98 web server traces.

Description	Default	Range
Number of Pages	4923	-
Request Rate Scaling Factor	1	[0.25-128]
Service Rate (pages/sec)	10	[2.5-40]
Mean Relative Deadline (sec)	60	[10-240]

TABLE II
WORKLOAD PARAMETERS AND SETTINGS

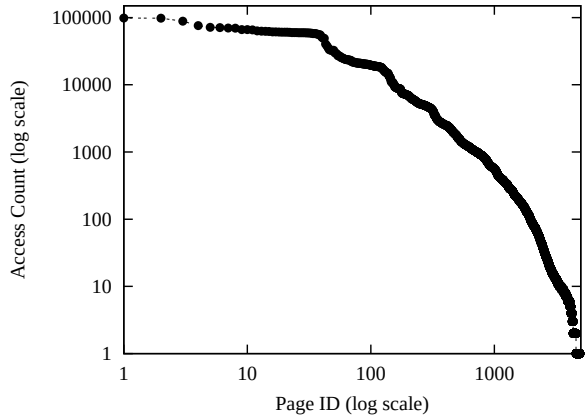
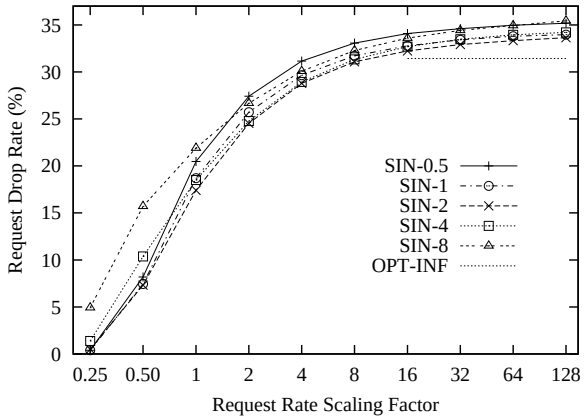


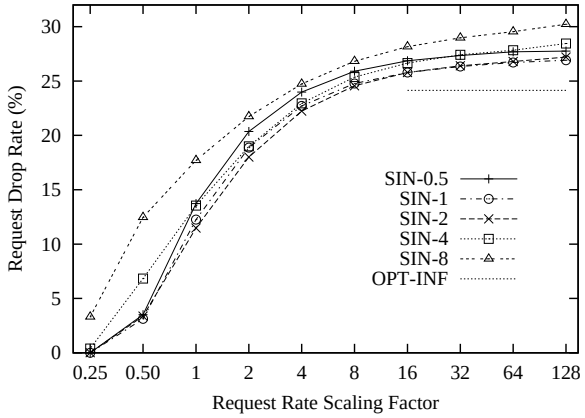
Fig. 4. Page Access Counts for the Day-38 Trace (4923 Pages)

rate scaling factor f. The inter-arrival time between two consecutive requests was set to the actual time logged in the trace divided by f. It is obvious that the higher the value of f, the heavier the workload. The service rate (i.e., capacity) of the broadcast channel is described in the number of pages that can be transmitted per second. The default service rate was set at 10 pages/sec. To model the time constraints of requests, each request was assigned a relative deadline d randomly generated based on a specified distribution (i.e., exponential, uniform, or fixed distributions) with the assigned mean value of the requested page. In the default setting, we differentiate the time requirements for different pages and assume the mean relative deadlines of requests for different pages are uniformly distributed between 0 and 120 seconds. Under this setting, the overall mean relative deadline of requests is 60 seconds. The workload parameters used in our experiments are summarized in Table II. Each simulation run started with an empty service queue. The first 1,000,000 requests were considered the start-up period, and performance statistics were collected for the subsequent 2,000,000 requests.

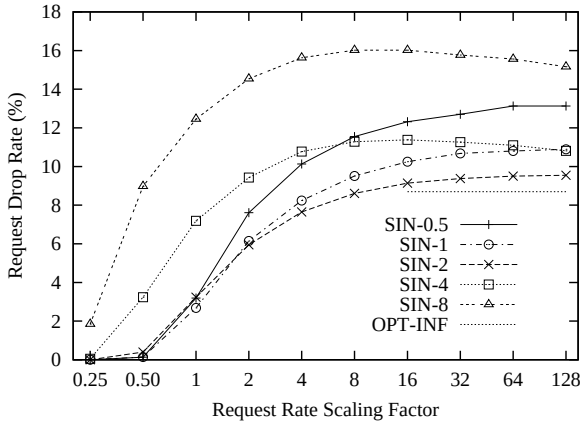
In the following, we first investigate the setting of parameter α in the SIN-α algorithm and examine its performance against the analytical optimum at high request rates. Next, we compare SIN-α with the state-of-the-art on-demand scheduling algorithms under various workloads. Finally, we evaluate the scheduling complexity of SIN-α.



(a) Exponential Deadlines



(b) Uniform Deadlines



(c) Fixed Deadlines

Fig. 5. Drop Rates of SIN-α for Different α Values

B. Impact of α Values and Optimality for SIN-α

Recall that in SIN-α, $\alpha \geq 0$ is a factor rating the relative importance of productivity and urgency for candidate data items (i.e., web pages). This set of experiments compares SIN-α against the analytical optimum for different

α values. To facilitate performance analysis, the mean relative deadlines for all pages were set at 60 seconds. Figure 5 shows the request drop rates of SIN-α with different values of α. We can observe that the α value of 1 or 2 gives the best overall performance, but neither value consistently dominates the other. The performance difference between the α values of 1 and 2 is not very significant. Therefore, in the rest of this section, we shall report and compare the performance of SIN-1 only with existing scheduling algorithms. Note that the SIN-α algorithm would obtain even better performance if the value of α can be fine tuned for specific workloads.

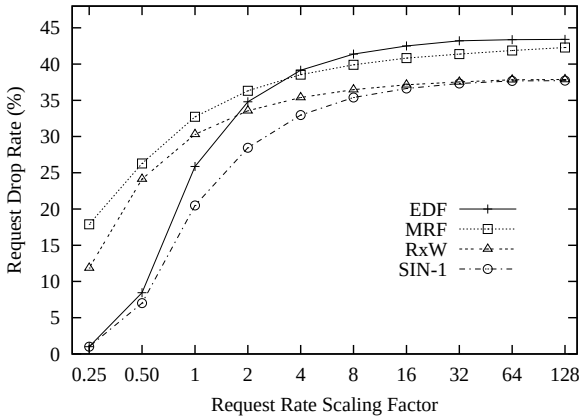
The analytical optimum when the request rate rises towards infinity, referred to as OPT-INF, is plotted in the right parts of Figure 5. OPT-INF serves as a lower bound on drop rate at high request rates. It can be seen that the proposed SIN-1 and SIN-2 algorithms perform very close to OPT-INF for scaling factors larger than 16.

C. Performance Comparison: Impact of Request Arrival Rate

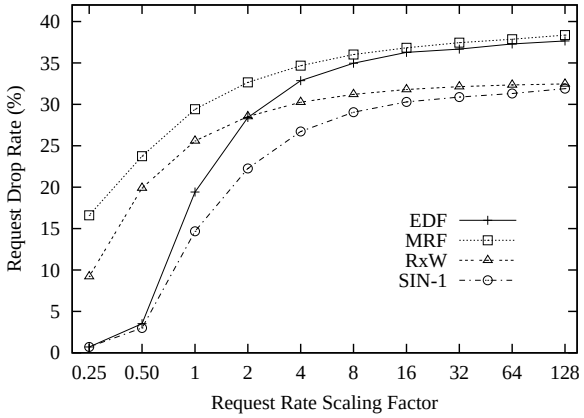
In this and subsequent subsections, we compare SIN-1 with the existing algorithms EDF, MRF, and the recently proposed R×W [3]. With R×W, the server broadcasts the page that has either a large number of pending requests or a long waiting time. The objective of R×W is to reduce the response time of requests. To simulate a more practical setting, the mean relative deadlines for different pages were randomly assigned between 0 and 120 seconds based on a uniform distribution. We first evaluate the scheduling algorithms with different request arrival rates. Figure 6 shows the request drop rates as a function of the scaling factor f.

Comparing different scheduling algorithms, the proposed SIN-1 algorithm performs the best throughout the tested range of scaling factor. The improvement of SIN-1 relative to EDF is up to 13.1%, 15.3%, and 38.0% for exponential, uniform, and fixed deadline distributions respectively and the improvement relative to MRF is at least 10.8%, 16.8%, and 49.8% respectively. Since MRF and R×W ignore the request deadlines, their drop rates are high even when the system is lightly loaded (see the left parts of Figure 6). In contrast, no requests are dropped by SIN-1 and EDF at low system loads. When the system is heavily loaded (see the right parts of Figure 6), SIN-1 performs substantially better than both MRF and EDF.

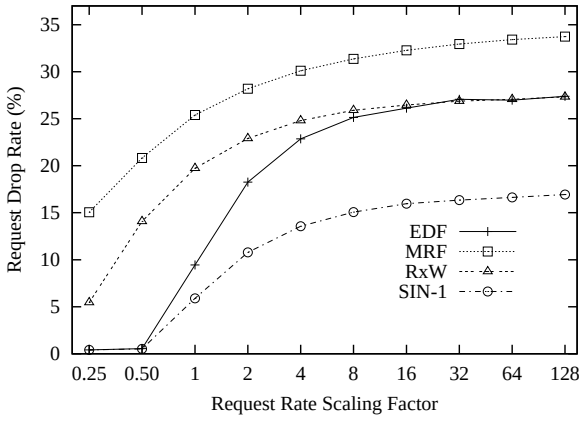
It is interesting to note that, among the three deadline distributions, the relative performance of MRF and R×W against EDF and SIN-1 is the worst when the relative deadline is fixed. This can be explained as follows. If the relative deadline spans over a range, a newly arrived request has a chance of overwriting the slack time of the requested item if there exist pending requests for the item already. Therefore, MRF and R×W, to some extent, take the urgency factor into consideration by first



(a) Exponential Deadlines



(b) Uniform Deadlines



(c) Fixed Deadlines

Fig. 6. Drop Rates for Different Arrival Rates

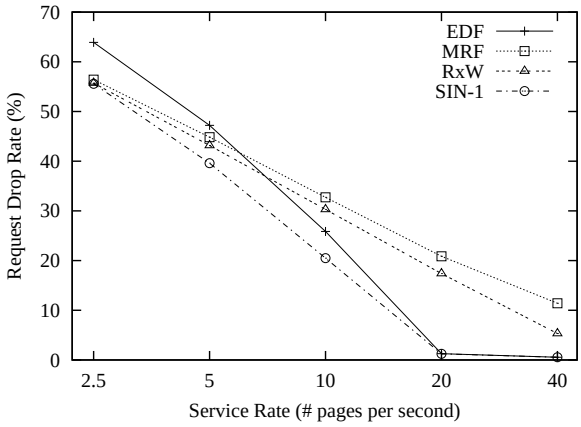
broadcasting the item with the largest number of pending requests and/or the longest waiting time. However, a new request never overwrites the slack time of the requested item under fixed deadline distribution. In this case, MRF and R×W completely ignore the urgency factor and performs much worse than SIN-1.

D. Performance Comparison: Impact of Service Rate

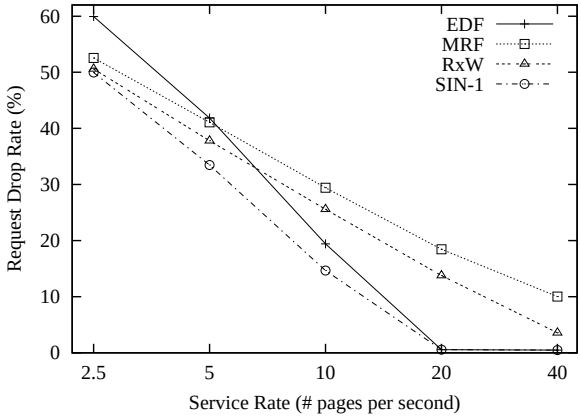
In this set of experiments, the scheduling algorithms are evaluated under different service rates. Figure 7 shows the results when the service rate is varied from 2.5 to 40. In general, MRF and EDF obtain good performance only under low service rates and high service rates respectively. When the service rate is low (i.e., 2.5 pages/sec), a significant portion of requests cannot be served on time. In this case, it is more important to improve the productivity of each broadcast item. Therefore, MRF outperforms EDF. On the other hand, when the service rate is high (i.e., 40 pages/sec), most requests can be served by their deadlines with a judicious schedule. In this case, it is more important to differentiate the requests by their deadlines. As a result, EDF performs better than MRF. By integrating the productivity and urgency, the proposed SIN-1 algorithm adapts well to a wide range of service rates and outperforms EDF and MRF for all deadline distributions examined. To achieve the same drop rate, EDF, MRF, and R×W would require much higher bandwidth than SIN-1. For example, as shown in Figure 7(c), to achieve the drop rate of 10%, MRF, R×W, and EDF require service rates of 39, 20, and 10 pages/sec respectively, whereas SIN-1 needs a service rate of 8.5 pages/sec only. This implies SIN-1 can save more than 75% bandwidth against MRF, 50% against R×W, and 15% against EDF.

E. Performance Comparison: Impact of Relative Deadline

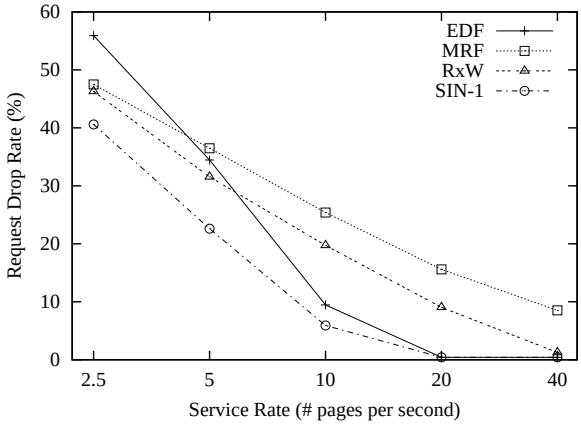
The lifetime of requests is a key parameter of time-critical applications. In this set of experiments, we examine the performance of the scheduling algorithms with respect to different relative deadlines. From Figure 8, it is clearly seen that SIN-1 consistently outperforms the other algorithms for all the deadlines tested. The flexibility of scheduling reduces with increasing urgency of requests, i.e., on average, fewer requests are served by a broadcast instance. Therefore, the workload of the system, to some extent, increases with decreasing relative deadlines. Due to the explanation in Section VI-D, MRF outperforms EDF under short relative deadlines (see the left parts of Figure 8), and EDF outperforms MRF under long relative deadlines (see the right parts of Figure 8).



(a) Exponential Deadlines



(b) Uniform Deadlines

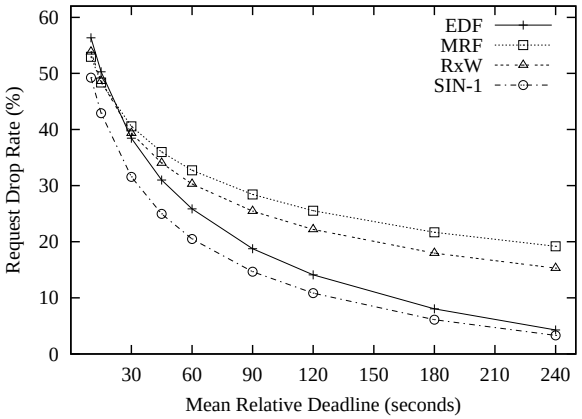


(c) Fixed Deadlines

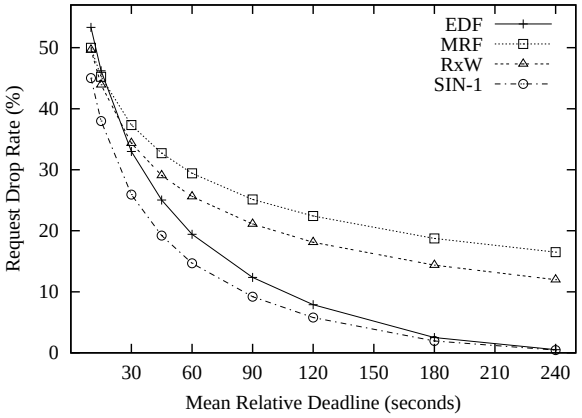
Fig. 7. Drop Rates for Different Service Rates

F. Evaluation of Scheduling Complexity

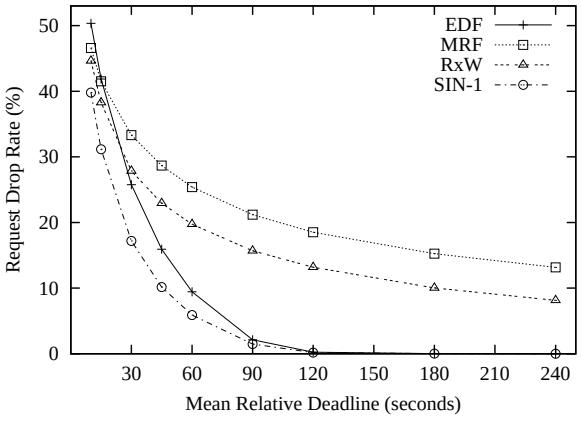
Scheduling complexity estimates the time to make scheduling decisions. It is desired to be low such that a scheduling decision can be made within a broadcast tick. This becomes more important with growing network



(a) Exponential Deadlines



(b) Uniform Deadlines



(c) Fixed Deadlines

Fig. 8. Drop Rates for Different Relative Deadlines

bandwidth since a higher bandwidth implies a shorter broadcast tick. The number of item entries traversed to make a scheduling decision is taken as a measure of scheduling complexity. Figure 9 shows the worst and average performance of two different implementations of SIN-1: a straightforward implementation that goes through all items with pending requests at each broadcast tick (referred to as *naive*), and the S-list/N-list based implementation proposed in Section IV-A (referred to as *proposed*). As can be seen, the scheduling complexity of the naive implementation increases rapidly with request rate. On the other hand, the scheduling complexity of the proposed implementation grows much slower compared to that of the naive implementation. Its average performance is well below 100 for all request rates examined, indicating good system scalability.

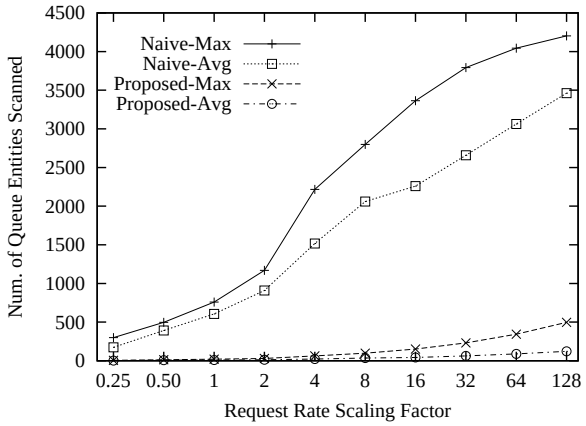


Fig. 9. Scheduling Complexity

VII. Conclusions and Future Work

On-demand broadcast has attracted a lot of attention due to its scalability to user population and adaptiveness to dynamic user access patterns. The scheduling problem for time-critical on-demand broadcast is becoming increasingly important with the rapid growth of time-critical information services and emerging applications such as mobile location-based services. A new scheduling algorithm called SIN-α that integrates the urgency and productivity of items has been proposed in this paper. Moreover, an analytical model has been developed to investigate the optimal broadcast schedule in terms of request drop rate when the request arrival rate rises towards infinity. The analytical results have been used as a yardstick in our experimental evaluations. It can also be employed to facilitate bandwidth allocation in planning and provisioning of on-demand broadcast. To the best of our knowledge, this is the first study to investigate the problem of scheduling time-critical on-demand broadcast.

Trace-driven simulation experiments show that the proposed SIN-α algorithm considerably outperforms existing algorithms over a wide range of workloads and approaches the analytical optimum when the request arrival rate is high. An efficient implementation of SIN-α

has been proposed to reduce the scheduling complexity and improve system scalability. In summary, SIN-α is an effective yet simple scheduling algorithm and can be used in practical time-critical on-demand broadcast systems such as satellite-based content distribution networks and wireless Internet.

The work reported in this paper is a first step to time-critical on-demand broadcast scheduling. There are many research issues that deserve further work. Ongoing research efforts include investigating the optimal setting for the weight α in SIN-α by an analytical study and building a prototype in a wireless LAN environment. We are going to develop energy-efficient scheduling algorithms for mobile computing applications to strike a balance between scheduling quality and mobile clients' energy consumption. This paper assumed the data to be broadcast is available on the broadcasting server; in practice, the data to be broadcast could reside on some remote servers, for which we plan to develop push-caching techniques to speed up data retrieval and facilitate broadcast scheduling. Another direction for future research is to investigate *soft* deadlines where the value of broadcasting an item is represented by a function of latency.

REFERENCES

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communications environments. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 199–210, San Jose, CA, May 1995.

[2] S. Acharya and S. Muthukrishnan. Scheduling on-demand broadcasts: New metrics and algorithms. In *Proc. MobiCom'98*, pages 43–54, Oct. 1998.

[3] D. Aksoy and M. Franklin. R×W: A scheduling approach for large-scale on-demand data broadcast. *IEEE/ACM Trans. on Networking (TON)*, 7(6):846–860, Dec. 1999.

[4] M. Arlitt and T. Jin. A workload characterization study of the 1998 world cup web site. *IEEE Network*, 14(3):30–37, May/June 2000.

[5] S. K. Baruah and A. Bestavros. Pinwheel scheduling for fault-tolerant broadcast disks in real-time database systems. In *Proc. ICDE'97*, pages 543–551, April 1997.

[6] A. Bestavros. AIDA-based real-time fault-tolerant broadcast disks. In *Proc. IEEE Real-Time Technology and Applications Symp. (RTAS'96)*, pages 49–58, June 1996.

[7] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proc. IEEE INFOCOM'99*, pages 126–134, March 1999.

[8] G. Buttazzo, M. Spuri, and F. Sensini. Value vs. deadline scheduling in overload conditions. In *Proc. IEEE Real-Time Systems Symp. (RTSS'95)*, pages 571–580, Dec. 1995.

[9] M.-S. Chen, K.-L. Wu, and P. S. Yu. Optimizing index allocation for sequential data broadcasting in wireless mobile computing. *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 15(1):161–173, Jan./Feb. 2003.

[10] A. Datta, D. E. VanderMeer, A. Celik, and V. Kumar. Broadcast protocols to support efficient retrieval from databases by mobile users. *ACM Trans. on Database Systems (TODS)*, 24(1):1–79, March 1999.

[11] J. Blazewicz et al. *Scheduling computer and manufacturing processes (2nd Edition)*. Berlin; New York: Springer, 2001.

[12] H. Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11:399–417, 1963.

[13] J. Fernandez and K. Ramamritham. Adaptive dissemination of data in time-critical asymmetric communication environments. In *Proc. Euromicro Real-Time Systems Symp.*, pages 195–203, 1999.

[14] C.-L. Hu and M.-S. Chen. Dynamic data broadcasting with traffic awareness. In *Proc. the 22nd IEEE Int. Conf. on Distributed Computing and Systems (ICDCS'02)*, pages 112–119, July 2002.

[15] Q. L. Hu, D. L. Lee and W.-C. Lee. Performance evaluation of a wireless hierarchical data dissemination system. In *Proc. MobiCom'99*, pages 163–173, August 1999.

[16] S. Jiang and N. H. Vaidya. Scheduling data broadcast to impatient users. In *Proc. MobiDE'99*, August 1999.

[17] D. L. Lee, W.-C. Lee, J. Xu, and B. Zheng. Data management in location-dependent information services. *IEEE Pervasive Computing*, 1(3):65–72, July-September 2002.

[18] V. Liberatore. Multicast scheduling for list requests. In *Proc. IEEE INFOCOM'02*, New York, NY, June 2002.

[19] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environments. *J. of ACM*, 20(1):46–61, 1973.

[20] D. A. Menasce. QoS issues in web services. *IEEE Internet Computing*, 6(6):72-75, Nov./Dec. 2002.

[21] A. Seifert and M. H. Scholl. A multi-version cache replacement and prefetching policy for hybrid data delivery environments. In *Proc. the 28th Int. Conf. on Very Large Data Bases (VLDB'02)*, pages 850–861, Hong Kong, August 2002.

[22] StarBand Communications. Website at http://www.starband.com/.

[23] Hughes Network Systems. DIRECWAY homepage. Website at http://www.direcway.com/.

[24] J. W. Wong. Broadcast delivery. *Proc. the IEEE*, 76(12):1566–1577, Dec. 1988.

[25] J. Xu, W.-C. Lee, and X. Tang. Exponential index: A distributed parameterized index for data on air. In *Proc. ACM/USENIX MobiSys'04*, June 2004.

[26] P. Xuan, S. Sen, O. Gonzalez, J. Fernandez, and K. Ramamritham. Broadcast on demand: Efficient and timely dissemination of data in mobile environments. In *Proc. IEEE Real-Time Technology and Applications Symp. (RTAS'97)*, pages 38–48, June 1997.

[27] WorldCup98 web site access logs, 1998. http://ita.ee.lbl.gov/html/contrib/WorldCup.html.

Appendix A: Proof of Theorem 1

Let d be a data item and α ($0 \le \alpha \le 1$) be the fraction of bandwidth allocated to d. We assume that d has been broadcast at tick 0, and consider d's broadcast schedule in a sufficiently long period starting from tick 1. Without loss of generality, we consider the interval from tick 1 to tick n/α, where n is some large positive integer. Since d is assigned a bandwidth share of α, it follows that d is broadcast $(n/\alpha) * \alpha = n$ times in the interval under investigation.

Suppose d is broadcast at times

$$l \le t_1 < t_2 < \cdots < t_n \le \frac{n}{\alpha} \cdot l.$$

Let

$$\tau_i = t_i - t_{i-1}, \qquad (i = 1, 2, \cdots, n)$$

be the duration between two consecutive broadcast instances t_i and t_{i-1} (for $i = 1$, define $t_0 = 0$).

According to Equation (1), the overall request drop rate during $[0, \frac{n}{\alpha} \cdot l]$ is given by

$$\frac{1}{\frac{n}{\alpha} \cdot l}\Big(\int_0^{\tau_1} F(\tau_1 - t)\, dt + \int_0^{\tau_2} F(\tau_2 - t)\, dt + \cdots + \int_0^{\tau_n} F(\tau_n - t)\, dt\Big). \quad (9)$$

We first prove that for any $i \neq j$,

$$\int_0^{\tau_i} F(\tau_i - t)\, dt + \int_0^{\tau_j} F(\tau_j - t)\big)\, dt \;\ge\; 2 \cdot \int_0^{\frac{\tau_i+\tau_j}{2}} F(\frac{\tau_i + \tau_j}{2} - t)\, dt.$$

Without loss of generality, suppose $\tau_i \ge \tau_j$. Since $F(x)$ is a non-decreasing function, we have

$$F(x) \;\ge\; F(x - \frac{\tau_i - \tau_j}{2}), \qquad (\frac{\tau_i + \tau_j}{2} \;\le\; x \;\le\; \tau_i).$$

Therefore,

$$\int_{\frac{\tau_i+\tau_j}{2}}^{\tau_i} F(x)\, dx \;\ge\; \int_{\frac{\tau_i+\tau_j}{2}}^{\tau_i} F(x - \frac{\tau_i - \tau_j}{2})\, dx,$$

and

$$\int_{\frac{\tau_i+\tau_j}{2}}^{\tau_i} F(x)\, dx \;\ge\; \int_{\tau_j}^{\frac{\tau_i+\tau_j}{2}} F(x)\, dx.$$

As a result,

$$\int_0^{\tau_i} F(x)\, dx + \int_0^{\tau_j} F(x)\, dx \;\ge\; 2 \cdot \int_0^{\frac{\tau_i+\tau_j}{2}} F(x)\, dx. \quad (10)$$

Since

$$\int_0^{\tau_i} F(\tau_i - t)\, dt = \int_0^{\tau_i} -F(\tau_i - t)\, d(\tau_i - t) = \int_{\tau_i}^{0} -F(x)\, dx \;=\; \int_0^{\tau_i} F(x)\, dx,$$

$$\int_0^{\tau_j} F(\tau_j - t)\, dt = \int_0^{\tau_j} -F(\tau_j - t)\, d(\tau_j - t) = \int_{\tau_j}^{0} -F(x)\, dx \;=\; \int_0^{\tau_j} F(x)\, dx,$$

and similarly,

$$\int_0^{\frac{\tau_i+\tau_j}{2}} F(\frac{\tau_i + \tau_j}{2} - t)\, dt \;=\; \int_0^{\frac{\tau_i+\tau_j}{2}} F(x)\, dx,$$

it follows from (10) that

$$\int_0^{\tau_i} F(\tau_i - t)\, dt + \int_0^{\tau_j} F(\tau_j - t)\, dt \;\ge\; 2 \cdot \int_0^{\frac{\tau_i+\tau_j}{2}} F(\frac{\tau_i + \tau_j}{2} - t)\, dt.$$

This implies the lowest drop rate is achieved when $\tau_1 = \tau_2 = \cdots = \tau_n$, since otherwise, we can obtain an equal or lower drop rate by replacing two different intervals τ_i and τ_j with $\frac{\tau_i+\tau_j}{2}$ and $\frac{\tau_i+\tau_j}{2}$. If $\tau_1 = \tau_2 = \cdots = \tau_n$, the overall drop rate (9) can be written as

$$\frac{1}{\frac{n}{\alpha} \cdot l} \cdot n \cdot \int_0^{\tau_n} F(\tau_n - t)\, dt \;=\; \frac{1}{\frac{n}{\alpha} \cdot l} \cdot n \cdot \int_0^{\tau_n} F(x)\, dx \;=\; \frac{\alpha}{l} \cdot \int_0^{\tau_n} F(x)\, dx. \quad (11)$$

Since $F(x)$ is a non-negative function, the overall drop rate (11) decreases with τ_n. Note that $n \cdot \tau_n = \tau_1 + \tau_2 + \cdots + \tau_n = t_n - t_0 \le \frac{n}{\alpha} \cdot l$. Therefore, the lowest drop rate is achieved when $\tau_1 = \tau_2 = \cdots = \tau_n = l/\alpha$.

Hence, the theorem is proven. □

Appendix B: Proof of Lemma 1

Assume on the contrary that there exist two items i and j such that $p_i > p_j$, $s_j \le L$ and $s_i > L$. Consider the set of inter-broadcast durations $(s'_1, s'_2, \cdots, s'_N)$ where $s'_i = s_j$, $s'_j = s_i$, and $\forall k \neq i, j, s'_k = s_k$. Let P be the drop rate of $(s_1, s_2, \cdots, s_N)$. The drop rate of $(s'_1, s'_2, \cdots, s'_N)$ is given by

$$\begin{aligned} P' &= P - p_i\eta_i - p_j\eta_j + p_i\eta'_i + p_j\eta'_j \\ &= P - p_i(1 - \frac{L}{2s_i}) - p_j\frac{s_j}{2L} + p_i\frac{s_j}{2L} + p_j(1 - \frac{L}{2s_i}) \\ &= P + (p_j - p_i)\big(1 - \frac{L}{2s_i} - \frac{s_j}{2L}\big) \\ &= P + (p_j - p_i)\frac{L(s_i - L) + s_i(L - s_j)}{2s_iL} \quad < P, \end{aligned}$$

which contradicts with the optimality of inter-broadcast durations $(s_1, s_2, \cdots, s_N)$.

Hence, the lemma is proven. □

Appendix C: Proof of Lemma 2

Let $(s_1, s_2, \cdots, s_N)$ be a set of inter-broadcast durations producing the lowest request drop rate. It follows from Lemma 1 that $\forall i \leq I$, $s_i \leq L$, and $\forall i > I$, $s_i > L$, where I is the identification item.

First, we prove that all non-infinity inter-broadcast durations in $s_{I+1}, s_{I+2}, \cdots, s_N$ are associated with data items of equal access probabilities. Assume on the contrary that there exist two non-infinity durations s_i and s_j $(j > i > I)$ such that $p_j < p_i$ (remember that item indexes are numbered in decreasing order of access probability). Consider the set of inter-broadcast durations $(s'_1, s'_2, \cdots, s'_N)$ where $s'_i = \frac{1}{\frac{1}{2s_i}+\frac{1}{2L}}$, $s'_j = \frac{1}{\frac{1}{s_j}+\frac{1}{2s_i}-\frac{1}{2L}}$, and $\forall k \neq i, j, s'_k = s_k$. It is easy to verify that $\sum_{k=1}^{N} \frac{1}{s_k} = \sum_{k=1}^{N} \frac{1}{s'_k}$, $s'_i > L$, and $s'_j > L$. Let P be the drop rate of $(s_1, s_2, \cdots, s_N)$. The drop rate of $(s'_1, s'_2, \cdots, s'_N)$ is given by

$$\begin{aligned}
P' &= P - p_i\eta_i - p_j\eta_j + p_i\eta'_i + p_j\eta'_j \\
&= P - p_i(1 - \frac{L}{2s_i}) - p_j(1 - \frac{L}{2s_j}) + \\
&\quad p_i(1 - \frac{L(\frac{1}{2s_i}+\frac{1}{2L})}{2}) + p_j(1 - \frac{L(\frac{1}{s_j}+\frac{1}{2s_i}-\frac{1}{2L})}{2}) \\
&= P + p_i(\frac{L}{4s_i} - \frac{1}{4}) + p_j(\frac{1}{4} - \frac{L}{4s_i}) \\
&= P + (p_i - p_j)\frac{L - s_i}{4s_i} \quad < P,
\end{aligned}$$

which contradicts with the optimality of inter-broadcast durations $(s_1, s_2, \cdots, s_N)$.

Moreover, it is easy to infer that the non-infinity inter-broadcast durations in $s_{I+1}, s_{I+2}, \cdots, s_N$ are associated with data items of access probability p_{I+1}. This is because otherwise, s_{I+1} must be infinity and therefore, exchanging a non-infinity duration with that of item $I+1$ would result in a lower drop rate.

So far, we have shown that given an optimal set of inter-broadcast durations $(s_1, s_2, \cdots, s_N)$, there exists an index J $(I \leq J \leq N)$ such that $\forall i \leq I$, $s_i \leq L$; $\forall I+1 \leq i \leq J$, $p_i = p_{I+1}$, $s_i > L$; and $\forall i > J$, $s_i = \infty$, where I is the identification item.

If $I = J$, $(s_1, s_2, \cdots, s_N)$ is an optimal set of durations following claims (i), (ii) and (iii).

Otherwise, if $I < J$, it is easy to show that the set of durations $(s'_1, s'_2, \cdots, s'_N)$ where

- $\forall i \leq I$, $s'_i = s_i$;
- $\forall I+1 \leq i \leq I + \lfloor \sum_{i=I+1}^{J} \frac{L}{s_i} \rfloor$, $s'_i = L$;
- $s'_{I+\lfloor \sum_{i=I+1}^{J} \frac{L}{s_i} \rfloor + 1} = \frac{1}{\sum_{i=I+1}^{J} \frac{L}{s_i} - \lfloor \sum_{i=I+1}^{J} \frac{L}{s_i} \rfloor \frac{1}{L}}$;
- $\forall I + \lfloor \sum_{i=I+1}^{J} \frac{L}{s_i} \rfloor + 2 \leq i \leq N$, $s'_i = \infty$,

also produces the lowest drop rate and satisfies claims (i), (ii) and (iii), where $I' = I + \lfloor \sum_{i=I+1}^{J} \frac{L}{s_i} \rfloor$ is taken as the identification item.

Hence, the lemma is proven. □

Appendix D: Proof of Lemma 3

We prove an even stronger claim: each set of values $(s_1, s_2, \cdots, s_I)$ where $\sum_{i=1}^{I} \frac{1}{s_i} = \frac{f}{l}$ and $\forall m+1 \leq i \leq I, s_i \leq L$ produces a value of (7) higher than or equal to that of durations $(s_1^*, s_2^*, \cdots, s_m^*, L, L, \cdots, L)$. Note that the definition of m ensures $\forall 1 \leq i \leq m$, $s_i^* \leq L$, and $f \geq \frac{I \cdot l}{L}$ implies $m \geq 1$.

The value of (7) for $(s_1^*, s_2^*, \cdots, s_m^*, L, L, \cdots, L)$ is given by

$$\frac{1}{2L(\frac{f}{l} - \frac{I-m}{L})}(\sum_{i=1}^{m} \sqrt{p_i})^2 + \sum_{i=m+1}^{I} \frac{p_i}{2}.$$

By applying the Lagrange multiplier method [12], the lowest value of (7) given $s_{m+1}, s_{m+2}, \cdots, s_I \leq L$ is:

$$\frac{1}{2L(\frac{f}{l} - \sum_{i=m+1}^{I} \frac{1}{s_i})}(\sum_{i=1}^{m} \sqrt{p_i})^2 + \sum_{i=m+1}^{I} \frac{p_i s_i}{2L}.$$

Thus, it is sufficient to prove

$$\frac{\left(\sum_{i=1}^{m} \sqrt{p_i}\right)^2}{2L(\frac{f}{l} - \sum_{i=m+1}^{I} \frac{1}{s_i})} + \sum_{i=m+1}^{I} \frac{p_i s_i}{2L} \geq \frac{\left(\sum_{i=1}^{m} \sqrt{p_i}\right)^2}{2L(\frac{f}{l} - \frac{I-m}{L})} + \sum_{i=m+1}^{I} \frac{p_i}{2} \tag{12}$$

$$\begin{aligned}
\iff & \sum_{i=m+1}^{I} p_i - \sum_{i=m+1}^{I} \frac{p_i s_i}{L} \leq \\
& \frac{\left(\sum_{i=1}^{m} \sqrt{p_i}\right)^2}{L(\frac{f}{l} - \sum_{i=m+1}^{I} \frac{1}{s_i})} - \frac{\left(\sum_{i=1}^{m} \sqrt{p_i}\right)^2}{L(\frac{f}{l} - \frac{I-m}{L})} \\
\iff & \sum_{i=m+1}^{I} p_i(1 - \frac{s_i}{L}) \leq \\
& \frac{\left(m - I + \sum_{m+1}^{I} \frac{L}{s_i}\right)\left(\sum_{i=1}^{m} \sqrt{p_i}\right)^2}{L^2(\frac{f}{l} - \frac{I-m}{L})(\frac{f}{l} - \sum_{i=m+1}^{I} \frac{1}{s_i})}.
\end{aligned}$$

Note that

$$\begin{aligned}
&\frac{\sum_{i=1}^{m+1}\sqrt{p_i}}{(\frac{f}{l}-\frac{I-m-1}{L})\sqrt{p_{m+1}}} > L \\
\iff\quad & \sqrt{p_{m+1}} < \frac{\sum_{i=1}^{m}\sqrt{p_i}+\sqrt{p_{m+1}}}{L(\frac{f}{l}-\frac{I-m}{L})+1} \\
\iff\quad & \sqrt{p_{m+1}} < \frac{\sum_{i=1}^{m}\sqrt{p_i}}{L(\frac{f}{l}-\frac{I-m}{L})} \\
\iff\quad & p_{m+1} < \frac{\big(\sum_{i=1}^{m}\sqrt{p_i}\big)^2}{L^2(\frac{f}{l}-\frac{I-m}{L})^2},
\end{aligned}$$

and

$$s_i \le L \iff \frac{1}{L} \le \frac{1}{s_i} \iff \frac{I-m}{L} \le \sum_{i=m+1}^{I}\frac{1}{s_i}.$$

Therefore,

$$\begin{aligned}
\sum_{i=m+1}^{I} p_i(1-\frac{s_i}{L}) &\le p_{m+1}\sum_{i=m+1}^{I}(1-\frac{s_i}{L}) \\
&\le \frac{\big(\sum_{i=1}^{m}\sqrt{p_i}\big)^2}{L^2(\frac{f}{l}-\frac{I-m}{L})^2}\sum_{i=m+1}^{I}(1-\frac{s_i}{L}) \\
&\le \frac{\big(\sum_{i=1}^{m}\sqrt{p_i}\big)^2\big(I-m-\sum_{i=m+1}^{I}\frac{s_i}{L}\big)}{L^2(\frac{f}{l}-\frac{I-m}{L})(\frac{f}{l}-\sum_{i=m+1}^{I}\frac{1}{s_i})}.
\end{aligned}$$

Since

$$\begin{aligned}
\frac{L}{s_i}+\frac{s_i}{L} \ge 2 \iff & \sum_{i=m+1}^{I}(\frac{L}{s_i}+\frac{s_i}{L}) \ge 2(I-m) \\
\iff & I-m-\sum_{i=m+1}^{I}\frac{s_i}{L} \le m-I+\sum_{i=m+1}^{I}\frac{L}{s_i},
\end{aligned}$$

it follows that

$$\begin{aligned}
\sum_{i=m+1}^{I} p_i(1-\frac{s_i}{L}) &\le \frac{\big(\sum_{i=1}^{m}\sqrt{p_i}\big)^2\big(I-m-\sum_{i=m+1}^{I}\frac{s_i}{L}\big)}{L^2(\frac{f}{l}-\frac{I-m}{L})(\frac{f}{l}-\sum_{i=m+1}^{I}\frac{1}{s_i})} \\
&\le \frac{\big(m-I+\sum_{m+1}^{I}\frac{L}{s_i}\big)\big(\sum_{i=1}^{m}\sqrt{p_i}\big)^2}{L^2(\frac{f}{l}-\frac{I-m}{L})(\frac{f}{l}-\sum_{i=m+1}^{I}\frac{1}{s_i})}.
\end{aligned}$$

Hence, the lemma is proven. □