Time-Critical On-Demand Data Broadcast: Algorithms, Analysis, and Performance Evaluation

Jianliang Xu, *Member*, *IEEE*, Xueyan Tang, *Member*, *IEEE*, and Wang-Chien Lee, *Member*, *IEEE*

Abstract—On-demand broadcast is an effective wireless data dissemination technique to enhance system scalability and deal with dynamic user access patterns. With the rapid growth of time-critical information services in emerging applications, there is an increasing need for the system to support timely data dissemination. This paper investigates online scheduling algorithms for time-critical on-demand data broadcast. We propose a novel scheduling algorithm called SIN-α that takes the urgency and number of outstanding requests into consideration. An efficient implementation of SIN-α is presented. We also analyze the theoretical bound of request drop rate when the request arrival rate rises toward infinity. Trace-driven experiments show that SIN-α significantly outperforms existing algorithms over a wide range of workloads and approaches the analytical bound at high request rates.

Index Terms—Mobile computing, on-demand data broadcast, scheduling, content delivery, time constraint.

1 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 [26] and DIRECWAY [27]), 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 item by a single transmission. This leads to a more efficient use of shared bandwidth, hence improving the system throughput and user-perceived response time [19], [30]. In general, there are two data broadcast approaches [8], [20]:[1] *Push-based broadcast* computes the broadcast program based on historical access statistics; *on-demand broadcast* schedules broadcast items on the fly based on current outstanding requests. While push-based broadcast is useful for certain applications (e.g., a small set of data items 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 [15], [24], [28]. 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. For example, in wireless financial services, many users are interested in the up-to-minute (or even "second") stock quotes in order to react to dynamic and rapid market developments. As another example, in wireless location-based services [18], the queried information (e.g., the local theaters) is valid only within a local area. When the mobile user moves away from the area, the information becomes invalid. In addition, a service level agreement (SLA) between a content/service provider and its users usually specifies the desired performance for Web requests, e.g., the response time of requests for CNN.com should not exceed 5 seconds [24]. In all the above cases, a *deadline* is associated with each request beyond which the serving of the request is useless (or less useful).

This paper focuses on on-demand data 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 data broadcast system is the *scheduling algorithm* used to select and broadcast requested items from outstanding requests. While there has been significant work on developing on-demand broadcast scheduling algorithms (e.g., [1], [2], [10], [29]), none of them has considered the time constraints associated with requests. On the other hand, although some time-critical scheduling algorithms have been proposed for unicast-based real-time systems and push-based broadcast systems (e.g., [11], [14]), they are not applicable or not effective to on-demand broadcast

1. A third approach is hybrid broadcast that combines on-demand broadcast with push-based broadcast.

- *J. Xu is with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, KLN, Hong Kong. E-mail: xujl@comp.hkbu.edu.hk.*
- *X. Tang is with the School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798. E-mail: asxytang@ntu.edu.sg.*
- *W.-C. Lee is with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802. E-mail: wlee@cse.psu.edu.*

Manuscript received 2 Nov. 2004; revised 15 Mar. 2005; accepted 26 Apr. 2005; published online 28 Nov. 2005.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0268-1104.

systems (see Section 2 for details). The objective of this paper is to develop 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 the design of on-demand broadcast scheduling algorithms. We propose a low-complexity scheduling algorithm, called SIN-α, for time-critical on-demand broadcast.
- We analyze the theoretical bound of request drop rate when the request arrival rate rises toward infinity. The analytical results are used as a yardstick in the experimental evaluation. They can also be employed to facilitate bandwidth provisioning in on-demand data broadcast systems.
- We conduct trace-driven simulation experiments to study the performance of the proposed scheduling algorithm over a wide range of workloads and show that SIN-α significantly outperforms existing algorithms and approaches the analytical bound at high request rates.

The rest of this paper is organized as follows: Section 2 summarizes the related work. The system model is described in Section 3. Section 4 presents the proposed scheduling algorithm, SIN-α, and discusses its implementation. Section 5 analyzes the theoretical bound of request drop rate when the request rate rises toward infinity. Section 6 experimentally compares the performance of the proposed algorithm against existing algorithms and the analytical bound. Finally, Section 7 concludes the paper with a brief discussion on future work.

2 Related Work

There has been much work on developing on-demand broadcast scheduling algorithms. Acharya and Muthukrishnan [1] introduced a new performance metric called stretch for variable-size data items and investigated several scheduling algorithms. Aksoy and Franklin [2] proposed a low-overhead and scalable scheduling algorithm called R×W. Datta et al. [9] took into consideration the energy saving issue in the design of on-demand broadcast systems. Liberatore [22] studied the scheduling algorithms for requests asking for a list of dependent items. Edmonds and Pruhs [10] proposed two constant approximation algorithms for scheduling variable-size data items. Hu and Chen [13] investigated dynamic traffic-aware scheduling algorithms. However, none of these algorithms has considered the time constraints of requests in making scheduling decisions.

Most existing studies on broadcast scheduling with time constraints investigated only periodic push-based broadcast [14], which is fundamentally different from on-demand broadcast in system architecture. Xuan et al. [31] 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. They 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.

A closely related area is task scheduling in real-time systems [7], [17], [23]. One of the most classical scheduling algorithms is EDF [23]. It offers optimal performance in light-loaded systems in terms of deadline missing rate. However, 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]. Buttazzo et al. [7] studied 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. Unfortunately, these existing scheduling algorithms are designed for a *unicast* environment where a newly arrived task cannot be combined with any existing tasks. In contrast, this paper focuses on a broadcast environment where a new request can be merged with existing outstanding requests if the same item is requested. These requests would be satisfied by a single broadcast of the item. This fundamental difference in system model gives rise to the development of new scheduling algorithms.

Other related work includes time-constrained transaction processing over *push-based* broadcast data [21], [25], where multiple versions of data items are broadcast to reduce the transaction abort rate and user-perceived response time. Schedules that minimize response time for *push-based* broadcast have been extensively studied in the algorithmic community (e.g., [5] and [16]). These studies complement 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. To the best of our knowledge, there has been no study on the scheduling algorithms for time-critical on-demand broadcast.

3 System Model

As shown in Fig. 1, we consider a satellite-based broadcast architecture that captures all essential components of a typical on-demand broadcast system [2], [3]. In this architecture, a large group of clients retrieve data items 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 considered useless 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. Active requests remain in the

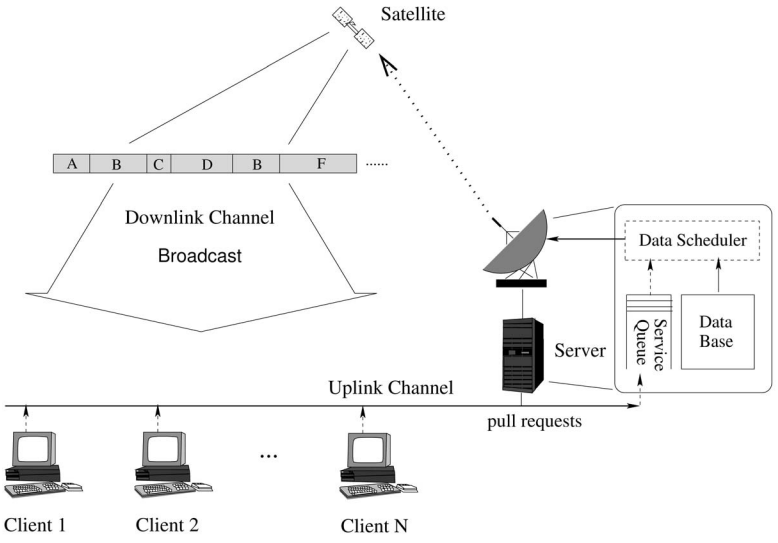


Fig. 1. A satellite-based broadcast architecture.

service queue until they are serviced or their lifetimes expire, whichever takes place earlier.

All data items are assumed to be locally available on the server. The server broadcasts data items based on a scheduling algorithm. The primary goal of a scheduling algorithm is to satisfy as many requests as possible. This can be 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. At each broadcast instance, the scheduler selects a new item from the active requests. The selected item is sent to the network controller for broadcast and the associated request(s) 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, they take equal time to broadcast). Note that variable-size data items can be easily handled by incorporating the factor of item size into our broadcast scheduling algorithm. In general, smaller items should be given higher priority than larger items in order to improve request drop rate.

4 Proposed SIN-α Algorithm and Its Implementation

In this section, we propose a new scheduling algorithm, called SIN-α. We start by illustrating the factors affecting the performance of time-critical broadcast scheduling. The time taken to broadcast each 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 broadcast. As will be shown in Section 6, 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,

- given two items with the same number of pending requests, the one with a closer deadline should be broadcast first to reduce request drop rate;
- given two items with the same deadline, the one with more pending requests should be broadcast first to reduce request drop rate.

Motivated by the above observations, we propose a new scheduling algorithm, called SIN-α (Slack time Inverse Number of pending requests).[2] 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 using the earliest deadline performs better than that with the mean/median deadline.

2. 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 (presented later in this section), and demonstrated good performance (see Section 6).

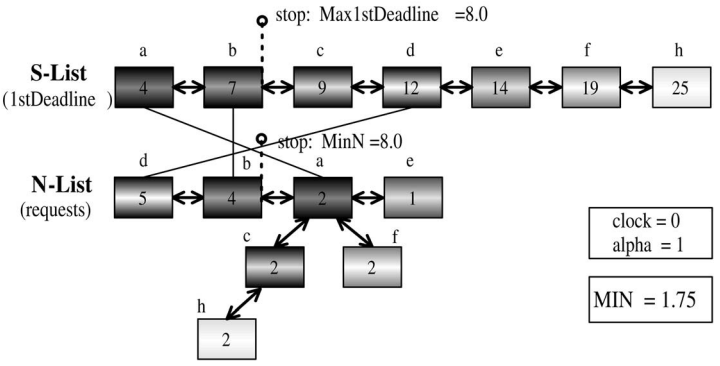


Fig. 2. Indexing structures of the service queue.

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 $\mathcal{O}(m)$, 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 [2].

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 Fig. 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 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*.

The two-lists indexing structure reduces 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, nonroot items in the N-list can be excluded from the search space. Second, the search space can be further pruned by sequentially searching the S-list and N-list in an alternate fashion. 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 $sin.\alpha$ value less 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 pseudocode 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.

Fig. 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 set $MIN = 2.4$, $MinN = \frac{4}{2.4} = 1.7$, and $Max1stDeadline = 4 \times 2.4 = 9.6$, meaning that an unexamined item would have a smaller $sin.\alpha$ value only if its *num* value exceeds 1.7 or its *slack* value is less than 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. Therefore, we update $MIN = 2.0$, $MinN = \frac{7}{2.0} = 3.5$, and $Max1stDeadline = 4 \times 2.0 = 8.0$. 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 $MinN = \frac{14}{1.75} = 8.0$ ($Max1stDeadline$ remains at 8.0 since *pn* becomes *nil*). The searching finishes here since the unexamined items (with $slack \geq 9$ and $num \leq 1$) do not have a $1stDeadline$ value less than $Max1stDeadline = 8.0$ and a *num* value greater than $MinN = 8.0$. Thus, there will not exist any item whose $sin.\alpha$ value is less than $MIN = 1.75$ (i.e., for item *b*). In total, we only need to examine three index entries to find the item to broadcast.

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 $sin.\alpha_{pn}$, the $sin.\alpha$ value of the item pointed by *pn*
7: **if** $sin.\alpha_{pn} < MIN$ **then** $MIN := sin.\alpha_{pn}$
8: **if** *ps* and *pn* refer to 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** $MinN := (\frac{ps \rightarrow 1stDeadline - clock}{MIN})^{\frac{1}{\alpha}}$

TABLE 1
Summary of Notations

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	broadcast tick (i.e., time taken to broadcast a single item)
$F(t)$	CDF of relative deadlines
$\eta(s_i)$	request drop rate of item i with an inter-broadcast duration s_i
$\eta(s_1, s_2, \cdots, s_N)$	total request drop rate of all items
M	the mean relative deadline in exponential distribution
L	the maximum relative deadline in uniform distribution
C	the relative deadline in fixed distribution

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 $sin.\alpha_{ps}$, the $sin.\alpha$ value of the item pointed by *ps*
17: **if** $sin.\alpha_{ps} <$ *MIN* then *MIN*:= $sin.\alpha_{ps}$
18: **if** *ps* and *pn* refer to 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**

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 the requests whose lifetimes will expire in the next broadcast tick.

5 Analytical Results

In this section, we analyze the theoretical bound of request drop rate when the request arrival rate rises toward infinity. The analytical results will be used as a yardstick in our experimental evaluation. The notations used in the analysis are summarized in Table 1.

Consider two consecutive broadcast instances of an item i at times 0 and τ (see Fig. 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 shorter than $(\tau - t)$. 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 arriving in interval $[0, \tau]$ is given by

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

where $F(t)$ is the CDF (cumulative distribution function) of relative deadlines, i.e., the probability that a relative deadline is shorter than 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. *To broadcast a data item for a given number of times between two broadcast instances, periodic broadcast offers the lowest request drop rate.*

Proof. Please refer to Appendix A (which can be found online at http://www.computer.org/tpds/archives.htm) for details. □

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

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

Therefore, the total request drop rate is given by

$$\eta(s_1, s_2, \cdots, s_N) = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda}\eta(s_i) = \sum_{i=1}^{N} \frac{p_i}{s_i}\int_0^{s_i} F(s_i - t)dt, \quad (2)$$

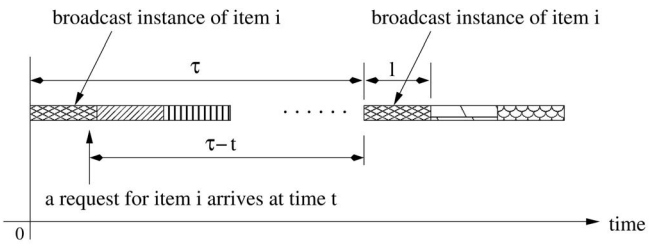


Fig. 3. System model.

where λ is the total request rate and λ_i and p_i are, respectively, the request rate and access probability of item i.

Given a broadcast tick of l, an interbroadcast duration of s_i implies a fraction $\frac{l}{s_i}$ of the bandwidth is used to broadcast item i. Therefore,

$$\sum_{i=1}^{N} \frac{l}{s_i} = 1 \Longrightarrow \sum_{i=1}^{N} \frac{1}{s_i} = \frac{1}{l}. \quad (3)$$

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

5.1 Exponentially Distributed Relative Deadlines

The CDF of an exponential distribution with a mean value 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 rewritten as

$$\eta(s_1, s_2, \cdots, s_N) = 1 - \sum_{i=1}^{N} \frac{p_i M(1 - e^{-\frac{s_i}{M}})}{s_i}. \quad (4)$$

Theorem 2. *Under exponentially distributed relative deadlines with a mean value of M, the total request drop rate is minimized when the set of interbroadcast durations $(s_1, s_2, \cdots, s_N)$ satisfies*

$$p_i(1 - \frac{s_i}{M} e^{-\frac{s_i}{M}} - e^{-\frac{s_i}{M}}) = K \qquad (i = 1, 2, \cdots, N), \quad (5)$$

for some constant K.

Proof. The theorem is proved by applying the Lagrange multiplier method [12]. See Appendix B (which can be found online at http://www.computer.org/tpds/archives.htm) for a sketch of the proof. □

Equation (5), together with constraint (3), can be solved numerically to compute the optimal bandwidth allocation and the lowest request drop rate (e.g., using a bisection method).

5.2 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) can be rewritten as

$$\eta(s_1, s_2, \cdots, s_N) = \sum_{i=1}^{N} p_i \eta(s_i), \quad (6)$$

where

$$\eta(s_i) = \begin{cases} \frac{s_i}{2L} & 0 < s_i \leq L, \\ 1 - \frac{L}{2s_i} & s_i > L. \end{cases}$$

Lemma 1. *Let $(s_1^*, s_2^*, \cdots, s_N^*)$ be a set of interbroadcast 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. Please refer to Appendix C (which can be found online at http://www.computer.org/tpds/archives.htm) for details. □

Without loss of generality, we number the items in decreasing order of access probability, i.e., $p_1 \geq p_2 \geq \cdots \geq p_N$. In the optimal bandwidth allocation, generally, the lower is the access probability, the less the bandwidth should be allocated. Lemma 1 implies that there exists an *identification item I* such that $\forall i \leq I$, $s_i^* \leq L$, and $\forall i > I$, $s_i^* > L$. Lemmas 2 and 3 further analyze the optimal bandwidth allocation.

Lemma 2. *There exists a set of interbroadcast durations $(s_1^*, s_2^*, \cdots, s_N^*)$ and an identification item I producing the lowest request drop rate such that*

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

Proof. Please refer to Appendix D (which can be found online at http://www.computer.org/tpds/archives.htm) for details. □

Lemma 3. *Assume a portion $f \geq \frac{I \cdot l}{L}$ of bandwidth is allocated to items $1, 2, \cdots, I$. The optimal set of interbroadcast durations minimizing the request drop rate of these items (from (6))*

$$\sum_{i=1}^{I} \frac{p_i s_i}{2L} \quad (7)$$

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

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

The lowest request drop rate for items 1 through I is given by

$$\eta(s_1^*, s_2^*, \cdots, s_I^*) = \frac{1}{2L(\frac{f}{l} - \frac{I-m}{L})} \left(\sum_{i=1}^{m} \sqrt{p_i} \right)^2 + \sum_{i=m+1}^{I} \frac{p_i}{2}. \quad (8)$$

Proof. Please refer to Appendix E (which can be found online at http://www.computer.org/tpds/archives.htm) for details. □

Now, assume a portion $f > 1 - \frac{l}{L}$ of bandwidth is allocated to items $1, 2, \cdots, I$. It follows from Lemma 2 that $s^*_{I+1} = \frac{l}{1-f} > L$ and $\forall i > I+1, s^*_i = \infty$. Hence, from (6), we have the request drop rate for items $I+1$ through N as follows:

$$\eta(s^*_{I+1}, s^*_{I+2}, \cdots, s^*_N) = p_{I+1}\left(1 - \frac{L}{2} \cdot \frac{1-f}{l}\right) + \sum_{i=I+2}^{N} p_i. \quad (9)$$

Thus, combining (8) and (9), the lowest request drop rate for items 1 through N can be computed by optimizing the following expression of f:

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

where

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

Hence, we have the following theorem.

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

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

where

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

5.3 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 rewritten as

$$\eta(s_1, s_2, \cdots, s_N) = \sum_{i=1}^{N} p_i \eta(s_i),$$

where

$$\eta(s_i) = \begin{cases} 0 & 0 < s_i \leq C, \\ 1 - \frac{C}{s_i} & s_i > C. \end{cases} \quad (10)$$

Theorem 4. *Under fixed relative deadline C, the total request drop rate is minimized when the set of interbroadcast durations $(s^*_1, s^*_2, \cdots, s^*_N)$ satisfies*

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

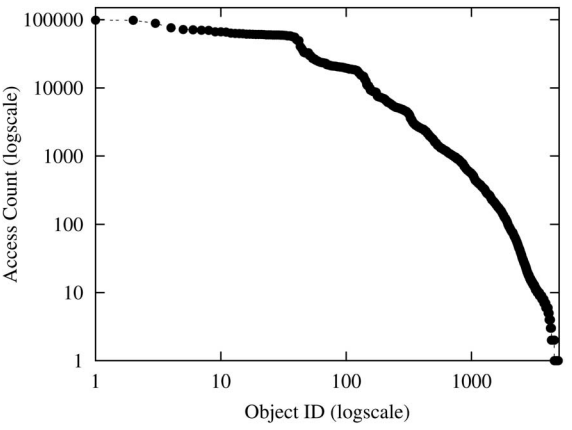


Fig. 4. Object access counts for the day-38 trace (4,923 objects).

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

$$\eta(s^*_1, s^*_2, \cdots, s^*_N) = p_{\lfloor y \rfloor + 1}(1 - y + \lfloor y \rfloor) + \sum_{i=\lfloor y \rfloor + 2}^{N} p_i.$$

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

In addition to serving as a yardstick in our performance evaluation, the analysis presented in this section can also be used to facilitate bandwidth provisioning in on-demand data broadcast systems. 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.

6 Performance Evaluation

6.1 Experimental Setup

A trace-driven simulator has been developed to evaluate the performance of proposed SIN-α algorithm. The simulator models the architecture shown in Fig. 1. Real traces collected from the World Cup '98 Website [32] were used to simulate the requests made by the clients. Similar performance trends were observed for different daily traces. Due to space limitations, we shall report only the experimental results of one trace (i.e., the day-38 trace) in this paper. The day-38 trace contains more than 7 million requests for 4,923 distinct Web objects (e.g., HTML pages and images).[3] The average request rate is 83 per second. The access counts of different objects sorted in descending order are shown in Fig. 4. It can be seen that the access pattern follows a Zipf-like distribution, which is consistent with the observation made in the literature [6]. To simulate different levels of workloads, we changed the time scale of the trace by introducing a *request rate scaling factor* f. The interarrival time between two consecutive requests in our experiments 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 data transmission rate of the broadcast channel is described in the number of objects that can be transmitted per second.

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

TABLE 2
Workload Parameters and Settings

Description	Default	Range
Number of Objects	4,923	-
Request Rate Scaling Factor	1	[0.25-128]
Data Transmission Rate of Broadcast Channel (objects/sec)	10	[2.5-40]
Mean Relative Deadline (sec)	60	[10-240]

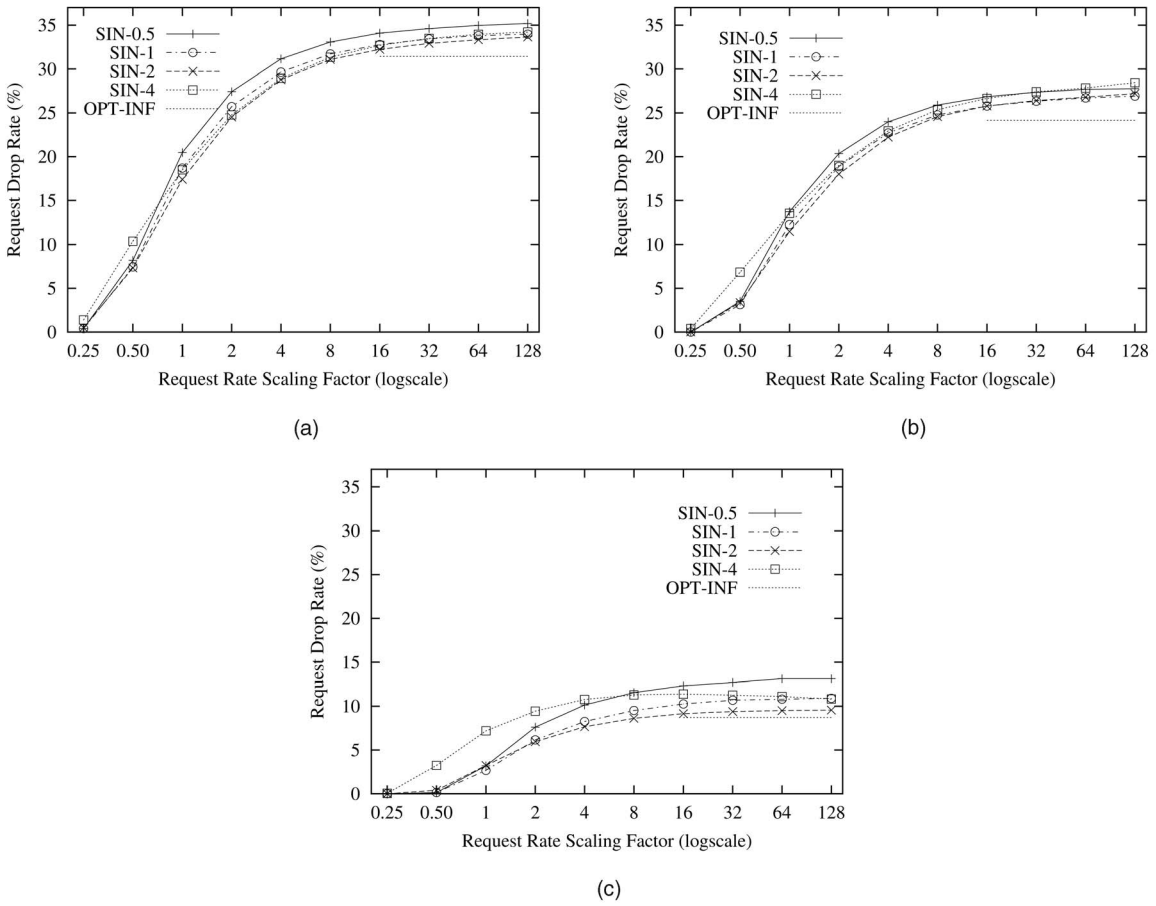


Fig. 5. Request drop rates of SIN-α for different α values. (a) Exponential deadlines, (b) uniform deadlines, and (c) fixed deadlines.

The default data transmission rate was set at 10 objects/sec (i.e., 0.1 second taken to broadcast an object). To model the time constraints of requests, each request was assigned a relative deadline randomly generated based on a specified distribution (i.e., exponential, uniform,[4] or fixed distribution) with the designated mean value. The workload parameters used in our experiments are summarized in Table 2. 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 bound at high request rates. Next, we compare SIN-α with the state-of-the-art scheduling algorithms under various workloads. Finally, we evaluate the scheduling complexity of SIN-α.

4. Given a mean value M, the uniform distribution sets relative deadlines randomly selected from a range of [0, 2M].

6.2 Impact of α Values

Recall that, in SIN-α, $\alpha \geq 0$ is a factor rating the relative importance of productivity and urgency for broadcasting candidate data items (i.e., Web objects). This set of experiments compares SIN-α with different α values against the analytical bound. To facilitate the comparison of the analytical bound, the mean relative deadlines for all objects were set at 60 seconds. As shown in Fig. 5, the request drop rate is not very sensitive to the setting of α. We can observe that α values of 1 and 2 give 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 remaining sections, we shall report and compare the performance of SIN-1 with existing scheduling algorithms.

The analytical lower bound on request drop rate when the request rate rises toward infinity is plotted in the right parts of Fig. 5 (referred to as OPT-INF). 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.

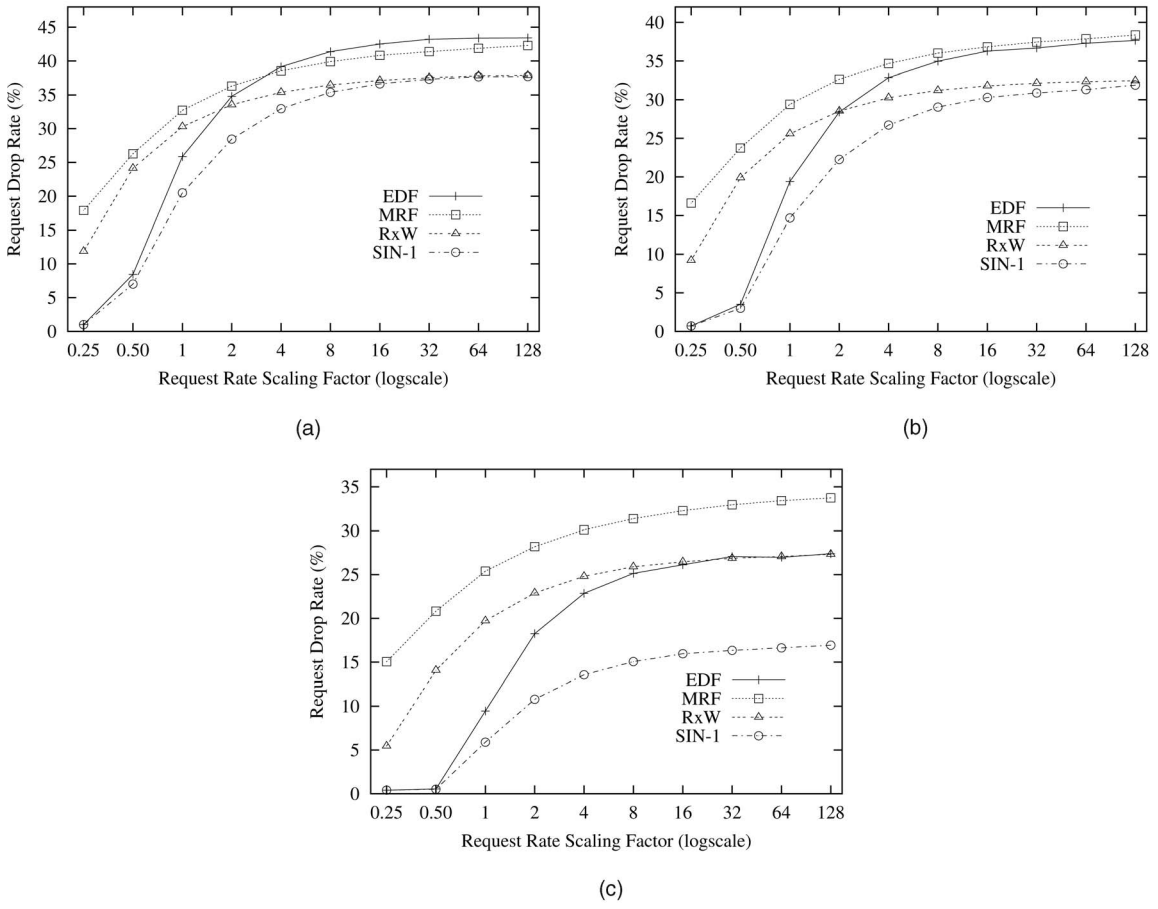


Fig. 6. Request drop rates for different arrival rates. (a) Exponential deadlines, (b) uniform deadlines, and (c) fixed deadlines.

6.3 Impact of Request Arrival Rate

In this and subsequent sections, we compare SIN-1 with the existing algorithms EDF, MRF, and R×W [2]. With R×W, the server broadcasts the object 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 Web objects were randomly assigned between 0 and 120 seconds based on a uniform distribution. Under this setting, the mean relative deadline of all requests is 60 seconds. The relative deadlines of requests for each object were randomly set based on its designated mean value following exponential, uniform, or fixed distribution. We first evaluate the scheduling algorithms under different request arrival rates. Fig. 6 shows the request drop rate as a function of the scaling factor f.

Comparing different scheduling algorithms, we can see that 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 percent, 15.3 percent, and 38.0 percent for exponential, uniform, and fixed deadline distributions, respectively, and the improvement relative to MRF is at least 10.8 percent, 16.8 percent, and 49.8 percent, 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 Fig. 6). In contrast, no request is dropped by SIN-1 and EDF at low system loads. When the system is heavily loaded (see the right parts of Fig. 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 SIN-1 is the worst when the relative deadline is fixed. This is because 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 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 perform much worse than SIN-1.

6.4 Impact of Data Transmission Rate

In this set of experiments, the scheduling algorithms are evaluated under different data transmission rates. Fig. 7 shows the results when the transmission rate is varied from 2.5 to 40 objects/sec. In general, MRF and EDF obtain good performance only under low transmission rates and high transmission rates, respectively. When the transmission rate is low (i.e., 2.5 objects/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 transmission rate is high (i.e., 40 objects/sec), most requests can be served by their deadlines with a careful schedule. In this case, it is more important to differentiate the requests by their deadlines. As a result, EDF performs better than

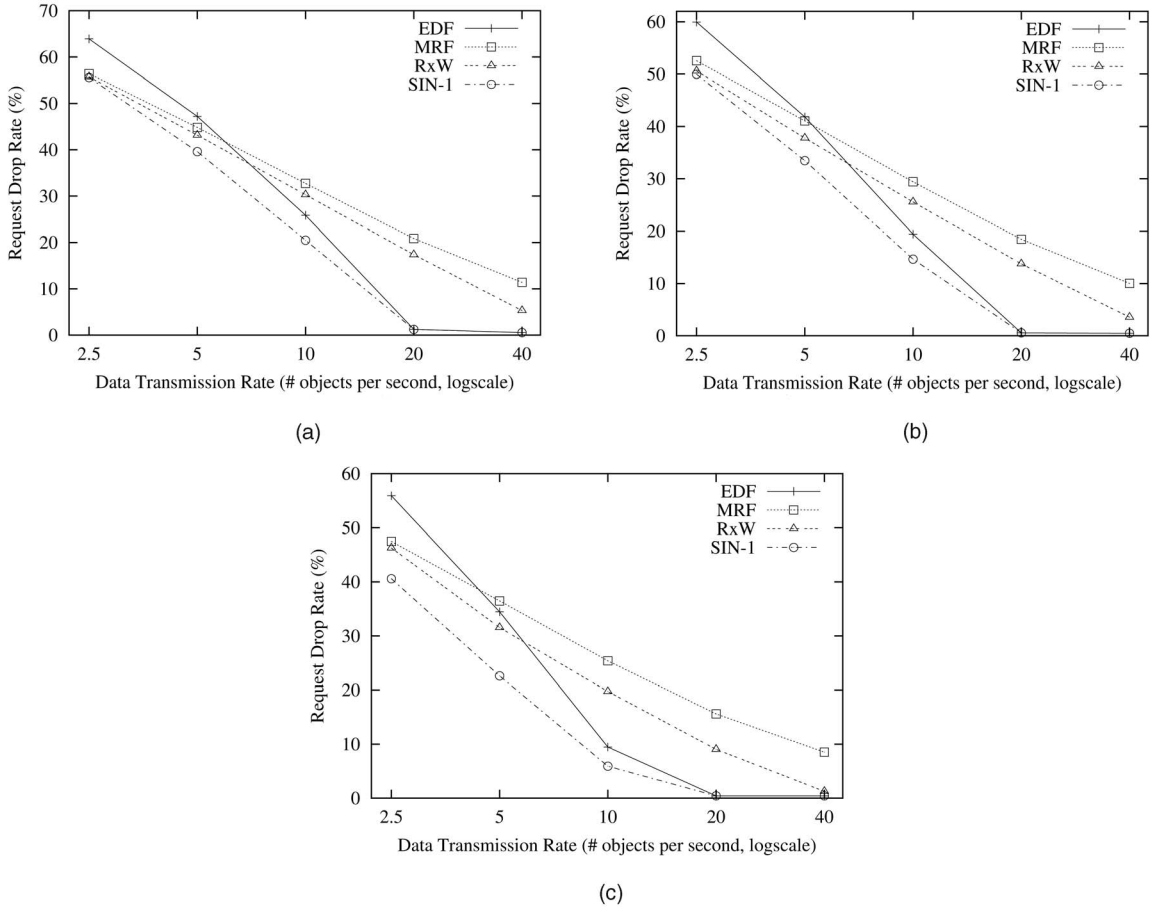


Fig. 7. Request drop rates for different data transmission rates. (a) Exponential deadlines, (b) uniform deadlines, and (c) fixed deadlines.

MRF. By integrating the productivity and urgency, the proposed SIN-1 algorithm adapts well to a wide range of transmission rates and outperforms EDF and MRF for all deadline distributions examined. To achieve the same request drop rate, EDF, MRF, and R×W would require much higher bandwidth than SIN-1. For example, as shown in Fig. 7c, to achieve a drop rate of 10 percent, MRF, R×W, and EDF require transmission rates of 39, 20, and 10 objects/sec, respectively, whereas SIN-1 needs a transmission rate of 8.5 objects/sec only. This implies SIN-1 can save more than 75 percent bandwidth against MRF, 50 percent against R×W, and 15 percent against EDF.

6.5 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 scheduling algorithms with respect to different relative deadlines. Fig. 8 shows that SIN-1 consistently outperforms the other algorithms over a wide range of deadlines. 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. For the reasons explained in Section 6.4, MRF outperforms EDF under short relative deadlines (see the left parts of Fig. 8) and EDF outperforms MRF under long relative deadlines (see the right parts of Fig. 8).

6.6 Evaluation of Scheduling Complexity

Scheduling complexity estimates the time taken to make scheduling decisions. The lower the complexity is, the more scalable is the scheduling algorithm. This becomes more important with growing network bandwidth since a higher bandwidth implies a shorter broadcast tick. In our experiments, the number of items examined to make a scheduling decision is taken as a measure of scheduling complexity. Fig. 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 implementation proposed in Section 4 (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.

7 Conclusions and Future Work

On-demand data 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 in emerging applications. A new scheduling algorithm called SIN-α that integrates the urgency and

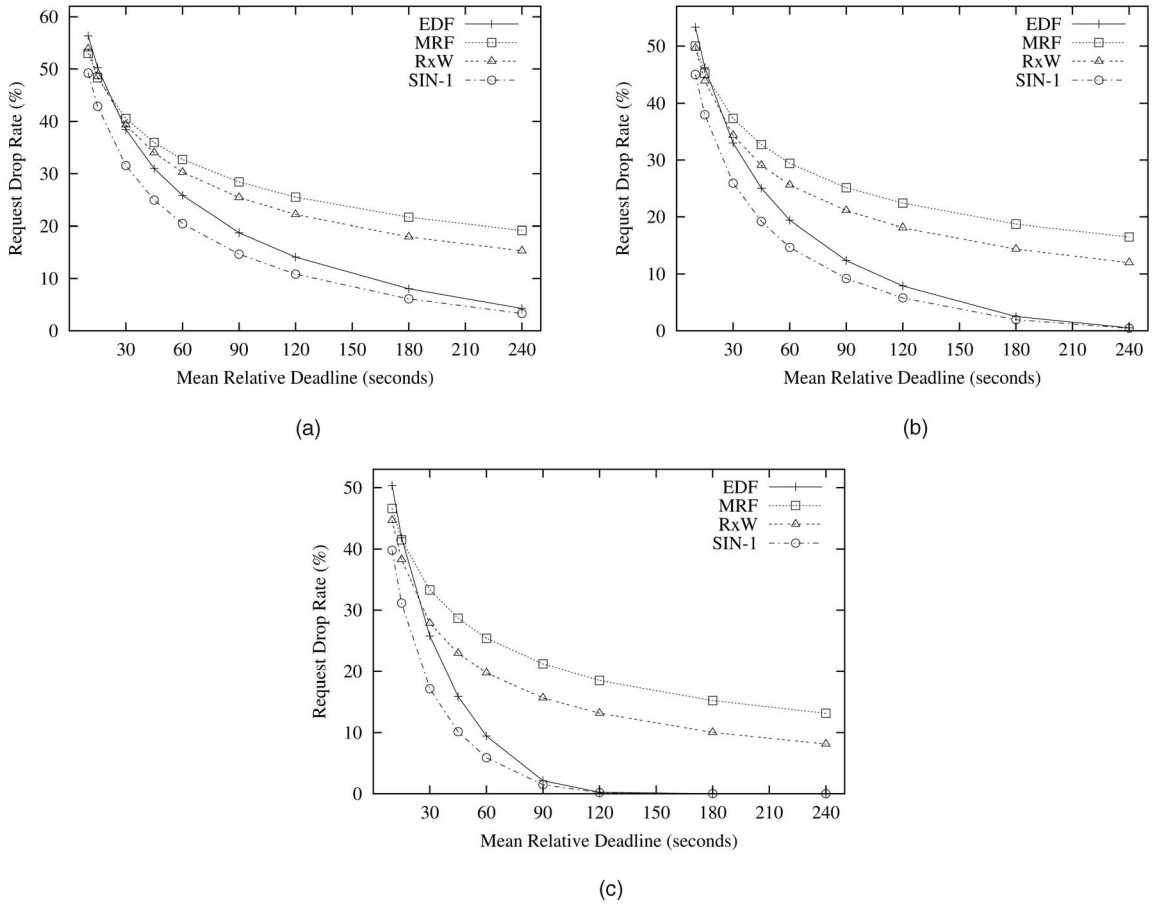


Fig. 8. Request drop rates for different relative deadlines. (a) Exponential deadlines, (b) uniform deadlines, and (c) fixed deadlines.

productivity of broadcast items has been proposed in this paper. An efficient implementation of SIN-α has been presented to reduce the scheduling complexity. Moreover, an analytical model has been developed to investigate the theoretical bound of request drop rate when the request arrival rate rises towards infinity. The analytical bound has been used as a yardstick in our experimental evaluation. It can also be employed to facilitate bandwidth provisioning in on-demand data broadcast systems.

Trace-driven simulation experiments have shown that the proposed SIN-α algorithm considerably outperforms existing algorithms over a wide range of workloads and approaches the analytical bound when the request arrival rate is high. In conclusion, 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.

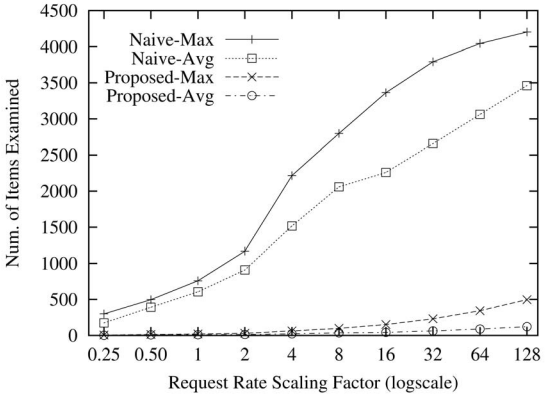


Fig. 9. Scheduling complexity.

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. 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-based caching techniques to speed up data retrieval and facilitate broadcast scheduling. We also plan to combine scheduling with indexing to improve energy efficiency for mobile clients. Another direction for future research is to investigate *soft* deadlines where the revenue of broadcasting an item is represented by a function of latency.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that improved the quality of this paper. Jianliang Xu's work was supported in part by grants from the Research Grants Council of Hong Kong, China (Grant Nos. HKBU 2115/05E, FRG/04-05/I-17, and FRG/04-05/II-21). Xueyan Tang's work was supported by a grant from Nanyang Technological University (Grant No. CE-SUG 1/04). Wang-Chien Lee's work was supported by the US National Science Foundation under Grant No. IIS-0328881.

REFERENCES

[1] S. Acharya and S. Muthukrishnan, "Scheduling On-Demand Broadcasts: New Metrics and Algorithms," *Proc. MobiCom '98,* pp. 43-54, Oct. 1998.

[2] D. Aksoy and M. Franklin, "R×W: A Scheduling Approach for Large-Scale On-Demand Data Broadcast," *IEEE/ACM Trans. Networking (ToN),* vol. 7, no. 6, pp. 846-860, Dec. 1999.

[3] D. Aksoy, M.J. Franklin, and S. Zdonik, "Data Staging For On-Demand Broadcast," *Proc. Conf. Very Large Data Bases '01,* pp. 571-580, Sept. 2001.

[4] M. Arlitt and T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site," *IEEE Network,* vol. 14, no. 3, pp. 30-37, May/June 2000.

[5] Z. Brakerski, A. Nisgav, and B.P. Shamir, "General Perfectly Periodic Scheduling," *Proc. 21st Ann. Symp. Principles of Distributed Computing (PODC '02),* pp. 163-172, July 2002.

[6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM '99,* pp. 126-134, Mar. 1999.

[7] G. Buttazzo, M. Spuri, and F. Sensini, "Value vs. Deadline Scheduling in Overload Conditions," *Proc. IEEE Real-Time Systems Symp. (RTSS '95),* pp. 571-580, Dec. 1995.

[8] M.-S. Chen, K.-L. Wu, and P.S. Yu, "Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing," *IEEE Trans. Knowledge and Data Eng.,* vol. 15, no. 1, pp. 161-173, Jan./Feb. 2003.

[9] A. Datta, D.E. VanderMeer, A. Celik, and V. Kumar, "Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users," *ACM Trans. Database Systems (TODS),* vol. 24, no. 1, pp. 1-79, Mar. 1999.

[10] J. Edmonds and K. Pruhs, "Broadcast Scheduling: When Fairness is Fine," *Proc. 13th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '02),* pp. 421-430, Jan. 2002.

[11] J. Blazewicz et al., *Scheduling Computer and Manufacturing Processes,* second ed. Springer, 2001.

[12] H. Everett, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Operations Research,* vol. 11, pp. 399-417, 1963.

[13] C.-L. Hu and M.-S. Chen, "Dynamic Data Broadcasting with Traffic Awareness," *Proc. 22nd IEEE Int'l Conf. Distributed Computing and Systems (ICDCS '02),* pp. 112-119, July 2002.

[14] S. Jiang and N.H. Vaidya, "Scheduling Data Broadcast to Impatient Users," *Proc. MobiDE '99,* pp. 52-59, Aug. 1999.

[15] M. Karlsson and C. Karamanolis, "Choosing Replica Placement Heuristics for Wide-Area Systems," *Proc. 24th IEEE Int'l Conf. Distributed Computing and Systems (ICDCS '04),* pp. 350-359, Mar. 2004.

[16] C. Kenyon, N. Schabanel, and N. Young, "Polynomial-Time Approximation Scheme for Data Broadcast," *Proc. 32nd Ann. ACM Symp. Theory of Computing (STOC '00),* pp. 659-666, May 2000.

[17] K. Lam and T. Kuo, *Real-Time Database Systems: Architecture and Techniques.* Kluwer Publisher, 2001.

[18] D.L. Lee, W.-C. Lee, J. Xu, and B. Zheng, "Data Management in Location-Dependent Information Services," *IEEE Pervasive Computing,* vol. 1, no. 3, pp. 65-72, July-Sept. 2002.

[19] K.C.K. Lee, H.V. Leong, and A. Si, "Semantic Data Broadcast for a Mobile Environment Based on Dynamic and Adaptive Chunking," *IEEE Trans. Computers,* vol. 51, no. 10, pp. 1253-1268, Oct. 2002.

[20] S. Lee, D.P. Carney, and S. Zdonik, "Index Hint for On-Demand Broadcasting," *Proc. IEEE Int'l Conf. Data Eng. '03,* pp. 726-728, Mar. 2003.

[21] V.C.S. Lee, J.K. Ng, J.Y.P. Chong, and K.-W. Lam, "Maintaining Temporal Consistency in Broadcast Environments," *Proc. IEEE Int'l Conf. Mobile Data Management (MDM '04),* pp. 284-292, Jan. 2004.

[22] V. Liberatore, "Multicast Scheduling for List Requests," *Proc. IEEE INFOCOM '02,* pp. 1129-1137, June 2002.

[23] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environments," *J. ACM,* vol. 20, no. 1, pp. 46-61, 1973.

[24] D.A. Menasce, "Qos Issues in Web Services," *IEEE Internet Computing,* vol. 6, no. 6, pp. 72-75, Nov./Dec. 2002.

[25] E. Pitoura and P.K. Chrysanthis, "Multiversion Data Broadcast," *IEEE Trans. Computers,* vol. 51, no. 10, pp. 1224-1230, Oct. 2002.

[26] StarBand Comm., http://www.starband.com/, 2006.

[27] Hughes Network Systems, DIRECWAY Homepage, http://www.direcway.com/, 2005.

[28] X. Tang and J. Xu, "QoS Aware Replica Placement for Content Distribution," *IEEE Trans. Parallel and Distributed Systems,* pp. 921-932, vol. 16, no. 10, Oct. 2005.

[29] J.W. Wong, "Broadcast Delivery," *Proc. IEEE,* vol. 76, no. 12, pp. 1566-1577, Dec. 1988.

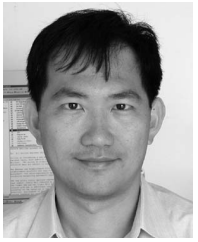
[30] J. Xu, W.-C. Lee, and X. Tang, "Exponential Index: A Distributed Parameterized Index for Data on Air," *Proc. ACM/USENIX MobiSys '04,* pp. 153-164, June 2004.

[31] P. Xuan et al., "Broadcast On Demand: Efficient and Timely Dissemination of Data in Mobile Environments," *Proc. IEEE Real-Time Technology and Applications Symp. (RTAS '97),* pp. 38-48, June 1997.

[32] World Cup 98 Web Site Access Logs, http://ita.ee.lbl.gov/html/contrib/WorldCup.html, 1998.

Jianliang Xu received the BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China, in 1998 and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2002. He is currently an assistant professor in the Department of Computer Science at Hong Kong Baptist University. His current research interests include mobile and pervasive computing, wireless sensor networks, and distributed systems. He has published more than 40 technical papers in these areas in leading journals and conferences, including ACM SIGMOD, MobiSys, IEEE ICDE, INFOCOM, TKDE, and TPDS. He has served as a session chair and program committee member for a number of international conferences including IEEE INFOCOM, IEEE MDM, and ACM SAC. He is a coeditor of the book *Web Content Delivery* (Springer). He is a member of the IEEE.

Xueyan Tang received the BEng degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 1998, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003. He is currently an assistant professor in the School of Computer Engineering at Nanyang Technological University, Singapore. He has served as a program committee member of IEEE INFOCOM '04 and WWW '05. He is a coeditor of the book *Web Content Delivery* (Springer). His research interests include Web and Internet (particularly caching, replication and content delivery), mobile and pervasive computing (especially data management and delivery), wireless sensor networks, and distributed systems. He is a member of the IEEE.



Wang-Chien Lee received the BS degree from the Information Science Department, National Chiao Tung University, Taiwan, the MS degree from the Computer Science Department, Indiana University, and the PhD degree from the Computer and Information Science Department, Ohio State University. He is currently an associate professor of computer science and engineering at Pennsylvania State University. Prior to joining the faculty of Penn State, he spent five plus years as a principal member of the technical staff at Verizon/GTE Laboratories, Inc. Dr. Lee's primary research interests lie in the areas of pervasive and mobile computing, data management, Internet technologies, wireless networks, and security. He has worked on building infrastructures to facilitate query processing in various wireless networks and pervasive computing systems (such as wireless broadcast systems, peer-to-peer systems, and sensor networks). He has served as a guest editor for several journal special issues on mobile database-related topics, including *IEEE Transactions on Computers*, *IEEE Personal Communications Magazine*, *ACM MONET*, and *ACM WINET*. He was a program committee cochair for the First International Conference on Mobile Data Access (MDA '99) and the International Workshop on Pervasive Computing (PC 2000). He is a member of the IEEE, the IEEE Computer Society, and the ACM.