Data & Knowledge Engineering 73 (2012) 23–42

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Editorial

Profit-based scheduling and channel allocation for multi-item requests in real-time on-demand data broadcast systems

Jingsong Lv [a,b], Victor C.S. Lee [a,*], Minming Li [a], Enhong Chen [b]

[a] Department of Computer Science, City University of Hong Kong and USTC-CityU Joint Advanced Research Centre, Suzhou, P. R. China
[b] School of Computer Science and Technology, University of Science and Technology of China and USTC-CityU Joint Advanced Research Centre, Suzhou, P. R. China

ARTICLE INFO

Article history:
Received 5 November 2010
Received in revised form 14 September 2011
Accepted 16 September 2011
Available online 2 October 2011

Keywords:
Data broadcast systems
Data scheduling
Channel allocation
Real-time systems

ABSTRACT

On-demand broadcast is a widely accepted approach for dynamic and scalable wireless information dissemination systems. With the proliferation of real-time applications, minimizing the request deadline miss ratio in scheduling multi-item requests has become an important task in the current architecture. In this paper, we prove the NP-hardness of broadcast scheduling of real-time multi-item requests in both single- and multi-channel environments. Furthermore, we propose two profit-based scheduling algorithms, PVC and SSA, for single- and multi-channel scheduling, respectively, both of which utilize our new concept "profit" of pending items and "opportunity cost" of pending requests. To the best of our knowledge, it is also the first time to introduce opportunity cost, which is derived from economics, into on-demand broadcast scheduling. Based on the scheduling result of PVC for pending requests, SSA is proposed to allocate selected items of scheduled requests to available channels. Finally, simulation results show great improvement in comparison with traditional algorithms. In general, PVC for single channel scheduling is superior to the best of other algorithms in terms of request deadline miss ratio. For multi-channel scheduling, SSA has larger advantage with increasing number of channels in terms of request deadline miss ratio than the best of other algorithms.

1. Introduction

With the rapid proliferation of software and hardware technology in mobile computing, information acquisition [1] has become increasingly important for emerging mobile applications. There are three major approaches to disseminating information: P2P, push-based data broadcast, and pull-based data broadcast. Recently, data broadcast approaches have attracted much attention due to their potential of satisfying all outstanding requests for the same data item with a single response.

The two principal data broadcast approaches, namely push-based and pull-based, are efficient in information dissemination under their respective application scenarios. The push-based approach periodically broadcasts a set of predetermined data items in a static manner. The broadcast program is based on some prior knowledge of the data access pattern of a community, regardless of individual information needs. As a result, it is suitable for certain applications with a relatively static data access behavior and small databases which admit short broadcast periods. In contrast, pull-based broadcast, also known as on-demand broadcast, is preferable for dynamic and large-scale data dissemination. However, on-demand broadcast requires an online scheduling algorithm which is hard to get optimal or near-optimal performance. As a matter of fact, on-demand broadcast scheduling with objective functions, such as minimizing average and maximum response time, is proved to be an NP-hard problem [2].

With the rapid development of real-time mobile applications and increasing population of mobile users, it can be envisioned that interactive information services have become popular. People submit time-critical requests for one or more dependent data items at a

* Corresponding author at: Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong. Tel.: +852 34428617; fax: +852 34420147.
E-mail address: csvlee@cityu.edu.hk (V.C.S. Lee).

0169-023X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2011.09.005

time. A time-critical request is associated with a deadline imposed by either the application or the user. The result of a request is useful only if all the requested data items can be received before the deadline. As a result, the primary goal is to minimize the number of requests missing their deadlines. For example, in a vehicular network application, drivers may want to know the traffic condition of multiple roads ahead in order to choose a route before arriving at the intersection. In a modern mobile application, users may also require multimedia data to support location-based services. To timely satisfy such multi-item requests for common data items in bandwidth-limited wireless environment, real-time on-demand data broadcast systems should be well designed, especially in terms of data scheduling algorithms.

In this study, we first show the NP-hardness of single channel broadcast scheduling of real-time multi-item requests. Then we propose an item level scheduling algorithm called Profit Versus Cost (PVC), which integrates three concepts in making scheduling decisions, namely "profit", "opportunity cost" and slack time. To evaluate the effectiveness of single channel scheduling of real-time multi-item requests, we design groups of simulation experiments for comparison with several traditional scheduling algorithms.

For multi-channel environments, we also prove NP-hardness of broadcast scheduling of real-time multi-item requests. Then, we utilize scheduling weights PVC proposed in single channel scheduling to support scheduling of real-time multi-item requests in multi-channel environment. Based on the scheduling part, we propose an allocation algorithm called SSA (Single Slot Allocation) to allocate some selected items from scheduled requests to available channels for one time slot. The allocation mainly utilizes three sets, namely, *WaitingQ*, *ServiceQ* and *ShelvedQ*, to perform iterative partitioning process for the pending requests with grouping process. After each iterative partitioning, an item can be selected and allocated to an available channel from a scheduled request based on the scheduling algorithm PVC. As shown by the simulation results, Algorithm SSA greatly outperforms other traditional algorithms.

The remainder of the paper is organized as follows. In Section 2, related work is reviewed. Section 3 shows the system model and some preliminaries. Section 4 focuses on single channel scheduling. NP-hardness of single channel scheduling is proved in Section 4.1. Two new concepts "profit" and "opportunity cost" are introduced in Section 4.2, followed by the proposed profit-based algorithm PVC. Section 4.3 analyzes the simulation results. Section 5 extends the single channel scheduling to multi-channel scheduling. The NP-hardness proof, the proposed algorithm SSA and its evaluation, are discussed in Sections 5.1, 5.2 and 5.3, respectively. Finally, Section 6 concludes the paper.

2. Related work

In data dissemination system environments, there are two main popular approaches to broadcasting information, namely, push-based broadcast and pull-based broadcast, also known as on-demand broadcast.

As push-based broadcast can utilize additional offline apriori information [3], it makes much more progress at an early time. Broadcast disk [4] is an efficient implementation for its hierarchical architecture to disseminate skewed data in accordance with the Square Root Rule (SRR), in comparison with the naive flat scheduling. In [5], a complementary approach is proposed to reduce empty slots in a broadcast cycle such that average access time is shorter than broadcast disk. As skewed scheduling can have better average access latency, it is applied to scheduling in multi-channel environments [6,7,8,9]. Motivated by its objective function of Average Expected Delay (AED), [6] and [9] utilize greedy strategies to partition data items into multiple channels, and the latter additionally applies dynamic programming techniques to the partition process. Recently, P. Yu et al. [9] extended SRR and proposed a new rule for near-optimal multi-channel scheduling, called Multi-channel Square Root Rule (MSRR) for non-uniform item size and non-uniform channel bandwidths. However, this approach is not scalable and not adaptable to dynamic system workloads and personal asymmetric demands.

On-demand broadcast is another well-known approach for data dissemination especially for scalable and dynamic broadcast [10]. RxW [11] is an effective on-demand scheduling algorithm for large-scale data broadcast, which integrates strengths of FCFS [12] and MRF [13], namely, waiting time and data access frequency. It approximates but is better than LWF [13] as it utilizes two corresponding queues to reduce running time. For reducing the average data access time, a Lazy Data Request (LDR) strategy [14] is proposed for message saving such that indexing can be used to reduce messages to help scheduling.

As both push-based and pull-based approaches have their own strengths and drawbacks [15,16], hybrid scheduling is regarded as a prospective approach to better scheduling. N. Saxena et al. [17] proposed a probabilistic hybrid scheduling, which probabilistically selects push operation or pull operation based on the present system statistics. Their results show that hybrid scheduling generally outperforms other purely push-based or pull-based algorithms in terms of access time. However, the above are all non-real-time scheduling.

For real-time on-demand scheduling, SIN [18] introduced the EDF strategy [19] to the traditional on-demand scheduling MRF [13], such that urgency and temperature of an item can be traded off for scheduling decisions. And for variable-length items (pages), X. Wu et al. [20] took item size (page size) into consideration for calculation of item level weights for preemptive scheduling. In addition to the above firm deadline scheduling, R. Dewri et al. [21] studied soft deadline scheduling using the evolution strategy which is a common method in artificial intelligence. About the hardness of such problems, J. Chang et al. [2] proved that broadcast scheduling with windows for single-item requests (firm deadline) is NP-hard.

Additionally, with the rapid development of complex applications [22,23], multi-item requests are becoming increasingly popular in data broadcast environments. QEM [24] opens up a new perspective in this field for non-real-time on-demand scheduling. It proposes a measure called Query Distance (QD), which shows the coherence degree of a request's data set in a schedule and indicates the access time of the request. And W. Sun et al. [25] have further added data replication to broadcast schedule of multi-item requests. Apart from the above non-real-time scheduling of multi-item requests, J. Chen et al. [26] focus on preemptive real-time on-demand scheduling of multi-item requests which, as a matter of fact, integrates urgency, item productivity (item access frequency) and request serving status factors into the scheduling. K. Lam et al. [27] studied the case of ordered data items in a mobile transaction (namely, a request) and

proposed a concurrency control method OUFO (Ordered Update First with Order) to solve the problem of post-updating consistency [28] and currency of data items. Besides this problem, they also used client caches to reduce data access delay. Note that the work above is for single channel scheduling.

For multi-channel scheduling of multi-item requests, K. Liu et al. [29] propose a client oriented scheduling algorithm COS to improve bandwidth utilization in on-demand broadcast. However, it is for non-real-time applications. C. Chen et al. [30] give the theoretical minimum number of channels (namely, bandwidth) for scheduling all given time-constrained multi-item requests and also propose a transformation-based data allocation algorithm to generate a periodic broadcast programme. However, as the pre-declared requests are obtained offline and the multi-item requests are assumed to be disjointed, namely, no overlap, the proposed algorithm is not suitable for online scheduling of on-demand requests.

Indexing is another important technique for client power saving by reducing client tuning time in on-demand broadcast scheduling. On-demand broadcast indexing was first discussed in [31] which mainly uses the estimation of broadcast times of data items. To adapt to the change of system workload, an adaptive on-demand broadcast indexing method [32] by dynamically adjusting the index and data organization is introduced and studied to reduce client power consumption at the cost of average access time. For the organizations of index segments and data segments, a global index model is depicted in [33], where two index schemes are discussed. The first index scheme is to separate index segments and data segments into index channels and data channels, respectively. The second index scheme interleaves index segments with data segments in the same channels. Indexing is orthogonal to our study, which focuses on data scheduling.

3. System architecture

The system consists of a number of clients and a single server (Fig. 1). Clients retrieve data items maintained by the server by sending requests through an uplink channel. One or more than one independent downlink channel is available to broadcast data items. The downlink channels typically have much greater communication capacity than the uplink channel, which is known as asymmetric communication [13,4,17]. A data item can be considered as a fixed-size page in the database; all data items share the same size. Given a single downlink channel with a fixed data transmission rate, namely, bandwidth $b = 1$, the time taken to broadcast a data item of a unit size is called a broadcast tick or a time slot. In this study, the size of an item is set as a unit size, namely, 1, and a time slot is regarded as a unit time. For multiple downlink channels, bandwidth of all channels is assumed to be the same.

Each client may request more than one data item at a time and each request has a release time and a deadline. A request is valid only if its deadline has not expired. Outstanding requests are queued up in the pending queue upon arrival at the server. The server, based on a certain scheduling algorithm, selects and retrieves the requested data items from the database for broadcasting. One broadcast data item has the potential to serve all requests pending for it. However, each client can retrieve a data item from only one of the channels at a time [34,35].

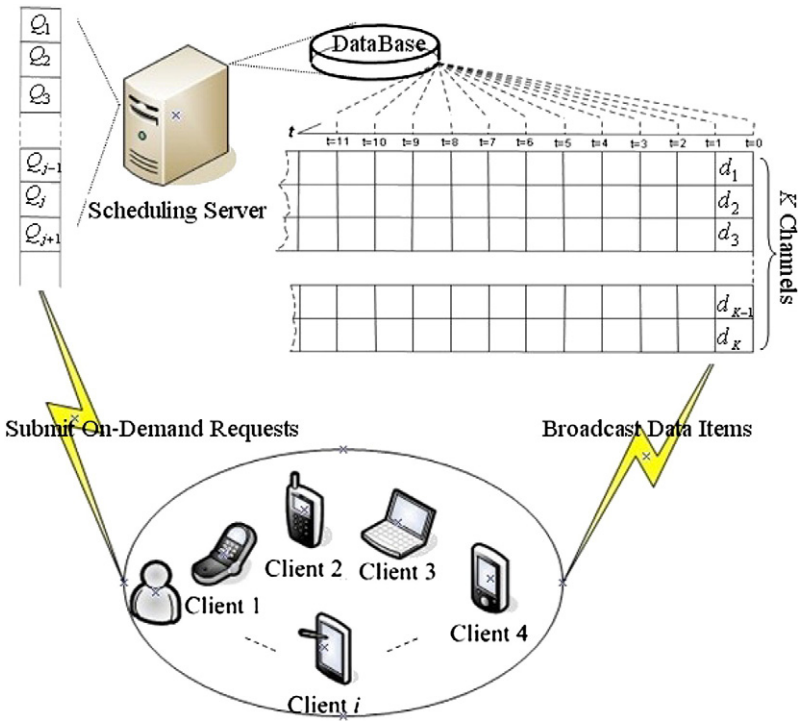

Fig. 1. System model.

The primary goal of the scheduling algorithm is to minimize the request deadline miss ratio or to maximize the number of requests that meet their deadlines. Lastly, a request can be satisfied only when all its required data items are received before its deadline. For details of notations used in this paper, please refer to Table 1.

4. Single channel scheduling

4.1. Hardness

In this section, we prove NP-hardness of real-time multi-item request scheduling. Although J. Chang [2] proved that broadcast scheduling of single-item requests with windows is NP-hard, which is similar to our problem, we use an easier way to prove it as our scheduling is for multi-item requests. First, we give the problem description, as follows.

Instance 1. Uniform multi-item single channel case: given m requests $Q_1, Q_2, \ldots, Q_m$, where each request Q_i has n_i data items of uniform size (size 1), and arrival time and deadline for each request are denoted as integers AT_i and Dl_i, respectively, and bandwidth of the only channel is $b=1$. Now given a bound $C \in Z^+$, denote the case as $\langle m, \{(n_i, AT_i, Dl_i) | i = 1, \ldots, m\}, C\rangle$. We ask if there is a schedule that satisfies at least C requests before their respective deadlines.

We show the NP-hardness of this problem by Theorem 1 and provide its proof.

Theorem 1. *Broadcast scheduling of multi-item requests with deadline constraint in single channel environment is NP-hard.*

Proof. We construct a polynomial time reduction from an instance of clique $\langle G, k\rangle$ to an instance of target special case $\langle m, \{(2,0,k) | i = 1, \ldots, m\}, \frac{k(k-1)}{2}\rangle$. Here, k is the size of the clique we want to find.

For a given undirected graph $G=\langle V, E\rangle$, the vertices in $V=\{A, B, \ldots\}$ are mapped to data items $d_A, d_B, \ldots$, respectively, and each edge (such as e_1) between any two vertices (such as A and B) is mapped to one request (Q_1) for the corresponding two data items (d_A and d_B). Arrival times and deadlines for all requests $Q_1, Q_2, \ldots$ are set to $AT=0$ and $Dl=k$. A simple example of the above transformation is shown in Fig. 2. In the examples presented in this paper, a request Q, which arrives at time AT with deadline Dl and requires an item set S, is represented as $S|[AT, Dl)$.

Apparently, the above transformation can be completed in polynomial time. Now, we show the graph has a clique of size k if and only if there is a schedule that can satisfy at least $\frac{k(k-1)}{2}$ requests for the single channel scheduling.

Suppose G has a clique $V' \subseteq V$ with size $|V'|=k$, then, there exist k corresponding data items that completely cover $\frac{k(k-1)}{2} = C$ different corresponding requests each of which is composed of any two out of the k data items based on the above transformation. In other words, given bound $C = \frac{k(k-1)}{2}$, there exists a schedule that satisfies at least C different requests.

Conversely, suppose given bound $C = \frac{Dl(Dl-1)}{2}$. There exists a schedule that satisfies at least C different requests before the deadline Dl. That is to say, Dl data items can satisfy C different requests before the deadline. Based on the transformation, $k=Dl$ corresponding vertices completely cover $C = \frac{Dl(Dl-1)}{2} = \frac{k(k-1)}{2}$ edges. Since any two different vertices from k different vertices can form at most $\binom{k}{2} = \frac{k(k-1)}{2} = C$ different edges only when the vertices and the edges form a complete graph. So, the corresponding k vertices and the corresponding C edges form a complete sub-graph of the undirected graph G. Namely, the undirected graph G has a clique of size k.

As finding a clique is NP-hard, Theorem 1 is proved. □

Table 1
Summary of notations.

Notation	Description
N	Number of data items in the database
K	Number of channels
m	Number of requests in the pending queue
f	Request arrival rate scaling factor
θ	Zipf distribution parameter
c	Number of clients
M	Total number of requests submitted by clients
k	Number of unserved data items in a request
s	Slack time of a request
Q/Q_j	A request or set of its unserved data items
n/n_i	Number of data items in some request
$\bar{n}$	Average number of items for all requests
b	Number of data items that can be broadcast within a time slot in single channel environment
b_i	Number of data items that can be broadcast within a time slot on channel i in K-channel environment
AT_i	Arrival time of a request
Dl_i	Deadline of a request
LAX_MIN	Lower bound of range of request deadline scaling factor
LAX_MAX	Upper bound of range of request deadline scaling factor

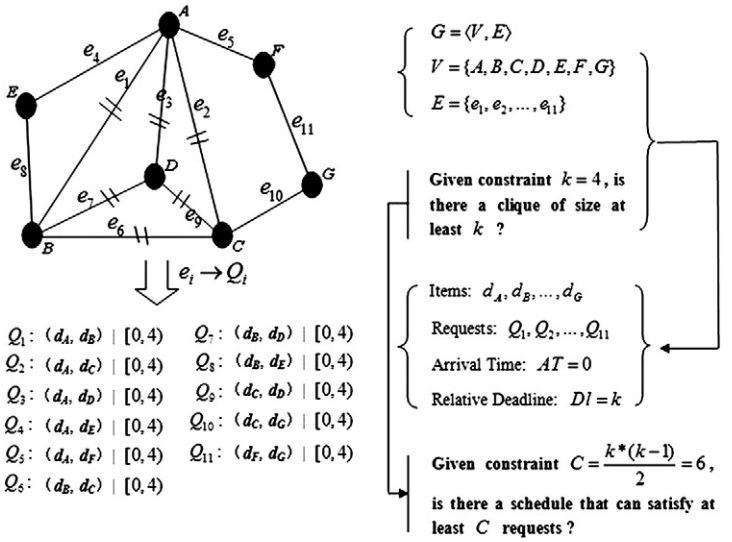

Fig. 2. An example of reduction process on Instance 1.

4.2. A profit-based algorithm

Since the scheduling problem is NP-hard, we propose a heuristic scheduling algorithm in this section. This algorithm integrates two new concepts, namely, profit and opportunity cost, for making scheduling decisions. An earlier version of this algorithm can be found in our previous work [36].

4.2.1. Profit

Previous research works have shown the effectiveness of item access frequency, also known as item productivity, in scheduling single-item requests in on-demand broadcast environments. Access frequency of a data item denotes the number of pending requests for it. It is obvious that broadcasting a data item with a higher access frequency can serve more requests in a single transmission.

However, in multi-item request environments, item access frequency exaggerates the contribution of broadcasting a data item to system performance as a request may not be satisfied by one data item. A request is satisfied when it receives all its required data items.

So it is insufficient to consider only the number of requests served in making a scheduling decision. Instead, we should consider the contribution of a pending data item to the system performance. Specifically, it is more profitable to broadcast a data item which can satisfy more requests or bring requests closer to completion. To quantify this idea, we propose a new concept, called "profit" to characterize the potential contribution of a pending item to system performance.

We first give definitions of some related concepts, as follows.

Definition 1. Slack time: the remaining time before the deadline of a request is said to be the slack time of the request. Notation s denotes the number of remaining time slots before the deadline of a request.

A request can meet its deadline only if the slack time is enough for broadcasting all its unserved data items. We define such a request as a live request. Note that in a data broadcast environment, a data item of a request may be served due to broadcast of the same data item of another request.

Definition 2. Live request: consider a request $Q(s, k)$, which consists of k unserved data items with slack time s. $Q(s, k)$ is said to be live if $s \geq k > 0$. Otherwise it is dead. When $k = 0$, the request is said to be complete as it has no unserved data item.

It is a routine task for the server to remove any complete or dead requests to ensure that all pending requests are live. Now we formally define "profit".

Definition 3. Relative profit: with respect to a request Q for n items with an unserved item set of cardinality k, the relative profit of an item d_i is defined as follows.

$$profit(d_i|Q) = \begin{cases} \frac{n-k+1}{n} & , \quad \text{if } d_i \in Q \\ 0 & , \quad \text{otherwise} \end{cases}$$

The relative profit denotes the contribution to one request via broadcasting of an item. In addition, the relative profit actually takes past scheduling decisions into account as $(n-k)$ is the number of items served to the request due to past scheduling decisions. Since a broadcast data item can serve all requests pending for it simultaneously, we can define an absolute profit for a data item as follows.

Definition 4. Absolute profit: the sum of relative profits of an item d_i with respect to all m requests in the pending queue is the absolute profit of d_i.

$$profit(d_i) = \sum_{j=1}^{m} profit(d_i|Q_j)$$

Absolute profit reflects the potential contribution to system performance via broadcasting of an item. See Example 1. Note that in the examples presented in this paper, a request Q, which arrives at time AT with deadline Dl and requires an item set S, is represented as $S|[AT, Dl)$.

Example 1. Q_1: $(d_1, d_2)|[0, \infty)$, Q_2: $(d_1, d_3)|[0, \infty)$, Q_3: $(d_4, d_5, d_6, d_7)|[0, \infty)$, Q_4: $(d_4, d_8, d_9, d_{10})|[0, \infty)$ and Q_5: $(d_4, d_{11}, d_{12}, d_{13})|[0, \infty)$.

Although access frequency of item d_4 is higher than that of item d_1, it is more profitable to broadcast item d_1 as it can bring requests (Q_1 and Q_2) closer to completion. So the contribution of broadcasting item d_1 to system performance is larger than d_4. Table 2 gives a more detailed comparison of item access frequency with absolute profit for Example 1.

Accordingly, the expected profit, or the average profit, for broadcasting one of the unserved data items in a request can be defined as follows. It can be observed that this metric can faithfully represent the contribution of serving a request to system performance. A request has a high average profit when its unserved data items are profitable. A data item is profitable if it is popular (hot) and its broadcast can satisfy more requests or bring more requests closer to completion.

Definition 5. Average profit: given a request Q with an unserved item set of cardinality k, the average profit for serving this request is defined as follows.

$$avgprofit(Q) = \frac{\sum_{d_i \in Q} profit(d_i)}{k}$$

4.2.2. Opportunity cost

We have defined the average profit for serving a request. However, like investment in economics, there is risk involved in serving any request. That is, the time investment on one request implies risk of missing the opportunity of serving other potential requests. This risk is called opportunity cost in economics. See Example 2.[1]

Example 2. $Q_1 \sim Q_6$: $(d_1, d_2, d_3)|[0, 5)$, Q_7: $(d_4)|[0, 3)$ and Q_8:$(d_5, d_6)|[0, 2)$.

If the scheduling decision is to broadcast a hot item, such as item d_1, to serve requests $Q_1 \sim Q_6$, the deadline of request Q_8 will be missed immediately and request Q_7 may lose the opportunity to get service as requests $Q_1 \sim Q_6$ may be served until satisfaction. However, if the scheduling decision is to broadcast item d_5 to serve request Q_8, the deadlines of other requests will not be missed until request Q_8 is satisfied if it is always in service before satisfaction.

To address this issue, we introduce the concept of *opportunity cost* in scheduling, which can interpret the above phenomenon. First we cite one definition of Opportunity Cost (OC) from [37]:"Opportunity cost is the value forgone by not using the transferred product in its next best alternative". Here, the next best alternative forgone implies that the decisions to choose different products are exclusive to each other. That means if two decisions are compatible with each other, neither has opportunity cost.

Now, we formally define "exclusive" and "compatible" as follows.

Definition 6. Relative compatible or exclusive requests: request $Q_j(s_j, k_j)$ is exclusive to request $Q(s, k)$ if $k + k_j - |Q \cap Q_j| > s_j$, otherwise Q_j is compatible with Q.

In other words, if Q_j is exclusive to Q, Q_j will be lost if Q is selected to be served until satisfaction. Note that if Q_j is exclusive to Q it does not imply that Q is also exclusive to Q_j.

In order to quantify values of other alternatives with respect to a scheduling decision, we introduce the *request expected revenue* and the *item expected revenue*. First, we define the request expected revenue which can be interpreted as the gain due to a request Q_j if the scheduling decision of serving request Q would be abandoned in favor of Q_j.

[1] Recall that a request Q, which arrives at time AT with deadline Dl and requires an item set S, is represented as $S|[AT,Dl)$.

Table 2
A comparison of item access frequency with absolute profit for Example 1.

d_i	Item access frequency	$profit(d_i)$
d_1	2	1
d_2, d_3	1	$\frac{1}{2}$
d_4	3	$\frac{3}{4}$
$d_5 \sim d_{13}$	1	$\frac{1}{4}$

Definition 7. Request expected revenue: given a request $Q_j(s_j, k_j)$, its expected revenue with respect to request $Q(s, k)$ is defined as follows.

$$E(Q_j|Q) = \begin{cases} \frac{k}{k_j} & , \text{ if } Q_j \text{ is exclusive to } Q \\ 0 & , \text{ otherwise} \end{cases}$$

Note that the request expected revenue not only captures the relationship between two requests (a request and another request exclusive to it), but also reflects their relative serving status. For instance, expected revenue of a request close to completion with respect to a request far from completion is higher than the vice versa. Next, we define item expected revenue as follows.

Definition 8. Item expected revenue: given an item d_i, its expected revenue with respect to request $Q(s, k)$ is the summation of expected revenue of all requests for d_i with respect to Q. That is:

$$E(d_i|Q) = \sum_{Q_j; d_i \in Q_j} E(Q_j|Q)$$

Item expected revenue $E(d_i|Q)$ can be interpreted as the gain due to the requests requiring an item d_i if the scheduling decision of serving request Q would be abandoned in favor of d_i. And the maximum item expected revenue of all items not required by Q is the opportunity cost of the scheduling decision of serving Q. Note that only items in those requests exclusive to Q can generate non-zero revenue. Accordingly, the next best alternative forgone is the item with the maximum expected revenue, which is the opportunity cost of Q, as defined below.

Definition 9. Opportunity cost: given a request $Q(s, k)$, the opportunity cost is defined as the maximum expected revenue with respect to Q of all items not requested by Q. That is:

$$C(Q) = \max_{d_i \notin Q} \{E(d_i|Q)\}$$

According to the above definitions, a comparison of item access frequency with opportunity cost for Example 2 is shown in Table 3. Based on the comparison, we can see that requests $Q_1 \sim Q_6$, including hot items, have higher opportunity cost while requests Q_7 and Q_8, including cold items, have lower opportunity cost. So a considerate scheduling decision should avoid high opportunity cost.

In economics or accounting, given the profit and cost, net profit which is equal to the (gross) profit minus cost, is a common measure of profitability. In line with our proposed concepts, we borrow this interpretation to construct the numerator of the scheduling weight. Intuitively, it should be more profitable to schedule a request with higher profit and lower opportunity cost. In addition, it is known that a request with less slack time (a closer deadline) should be scheduled with higher priority as it is more urgent [18,19]. So, we put slack time (s) in the denominator of the scheduling weight to reflect this relationship.

Based on the above discussion, we integrate the two newly proposed concepts and slack time into a scheduling weight as defined below.

Definition 10. Scheduling weight: the scheduling weight of a request $Q(s, k)$ is defined as follows.

$$PVC(Q) = \frac{avgprofit(Q) - C(Q)}{s(Q)}$$

Table 3
A comparison of item access frequency with opportunity cost for Example 2.

Q	Avg. item access frequency	$C(Q)$
$Q_1 \sim Q_6$	6	3
Q_7	1	$\frac{1}{2}$
Q_8	1	0

This PVC weight is used in our proposed item level scheduling algorithm, called Profit Versus Cost (PVC), as shown in Algorithm 1. The basic idea of Algorithm PVC is to first choose the request with the largest scheduling weight and then broadcast the item with the largest absolute profit in the selected request. The algorithm mainly includes three parts. The first part is to queue and preprocess newly arrived requests. The second part is to select an item and broadcast it. The third part is to update the pending queue constructed in the first part after broadcasting.

As the main part of the algorithm is Part 2, item selection and broadcast, we analyze the time complexity of this part as follows. The first sub-part of removing infeasible requests for lines 16–19 can be completed in time $O(m)$, where m is the number of requests in the pending queue. For the second sub-part of calculations of scheduling weights PVC for lines 21–42, the outer for loops of line 23 and line 38 are repeated at most m times in total and the inner for loops can be finished in time at most $O(n \cdot m)$. So the second sub-part can be completed in time $O(n \cdot m^2) = O(m^2)$. The following sub-part of item selection can be completed in time $O(n)$. To conclude, the time complexity of the algorithm is $O(m^2)$.

To reduce the running time of PVC, an efficient implementation is presented in Algorithm 1. Two simple data structures and a pruning technique are introduced to reduce the searching space. In other words, the number of requests to be examined while finding the request with the highest scheduling weight (*PVC* value) can be greatly reduced. The main idea is to avoid the computation of the opportunity cost of a request, C(Q), which is the most computational expensive part in the formula of scheduling weight indicated in Definition 10. We need to compute the opportunity cost of a request only if its $\frac{avgprofit(Q)}{s(Q)}$ value is larger than the current maximum *PVC* value. Otherwise, it is impossible for the request to have a *PVC* value larger than the current maximum. Thus, there is no need to compute its *PVC* value consisting of the opportunity cost. The details of our proposed efficient implementation of PVC are described below.

Two data structures, a *ProfitQ* and a *SlackQ,* are used to index the requests in the pending queue. In the *ProfitQ,* requests are sorted in descending order of average profit. In the *SlackQ,* requests are sorted in ascending order of deadline. These two data structures are updated incrementally upon arrival of a new request or broadcasting of a selected item. The searching is done by sequentially traversing *ProfitQ* and *SlackQ* once each. To prune the searching space, two values, P_{max} and S_{min}, are maintained in *ProfitQ* and *SlackQ,* respectively. P_{max} is the current maximum average profit of unexamined requests in *ProfitQ* and S_{min} is the current minimum slack time of unexamined requests in *SlackQ.* Let T denote the current maximum *PVC* value which is initialized to zero. *ProfitQ* is searched first. It is possible for the current request to have a *PVC* value larger than T only if its average profit exceeds $MinP = T \cdot S_{min}$. Only in this case we need to compute its *PVC* value. If the *PVC* value is larger than T, T is updated to the new *PVC* value. The searching of *ProfitQ* continues until all requests are searched or the condition indicated by *MinP* is violated (i.e., average profit of current request $\leq MinP$). Next, *SlackQ* is searched. Similarly, it is possible for the current request to have a *PVC* value larger than T only if its slack time is less than $MaxS = \frac{P_{max}}{T}$. Only in this case we need to compute its *PVC* value. If the *PVC* value is larger than T, T is updated to the new *PVC* value. The searching of *SlackQ* continues until all requests are searched or the condition indicated by *MaxS* is violated (i.e., slack time of current request $\geq MaxS$). At the end, the request with the maximum *PVC* value can be identified accordingly. In practice, the number of requests that have to be examined is much lower than the number of requests in the pending queue (i.e. m) because the value of *MinP* is increasing and that of *MaxS* is decreasing as the searching proceeds.

4.3. Performance evaluation

4.3.1. Algorithms and grouping

To better understand the characteristics of our proposed new concept "profit", we implement two other variations, PROFIT and PXW, to study its effectiveness. In addition, we implement several existing best-performing algorithms for comparison with our proposed scheduling algorithm PVC which also incorporates another new concept "opportunity cost" in making scheduling decisions. Based on factors considered in making scheduling decisions, the algorithms are categorized into 3 groups for better comparison and analysis.

Group A: MRF [13] and PROFIT. MRF schedules the item with the largest number of pending requests to ensure item productivity. PROFIT schedules the item with the largest absolute profit (see Definition 4). This group considers neither the deadline nor the waiting time of a request.

Group B: FCFS [12], RXW [11], LWF [13] and PXW. This group of algorithms considers request waiting time in making scheduling decisions. FCFS serves the request which has been waiting the longest time. In other words, requests are served in the same order as their arrival. Both RXW and LWF integrate item productivity and request waiting time into scheduling. LWF schedules the data item with the longest total waiting time, which is the sum of waiting time of all pending requests. RXW calculates the product of the waiting time of the oldest pending request for a data item and its item productivity. The data item with the largest value is scheduled. PXW simply replaces the item productivity in RXW by the absolute profit of a data item in calculating the product.

Group C: EDF [19], SIN [18] and PVC. All algorithms in this group consider request deadline. EDF is a well-known real-time scheduling algorithm. Its basic idea is to serve the most urgent request first, i.e. the request with the earliest deadline. SIN considers both request urgency and item productivity. The data item with the smallest SIN-value is scheduled first; SIN-value of an item is defined as the ratio of slack time of its parent request to its item productivity.

PVC is our proposed scheduling algorithm.

4.3.2. Experimental setup

Next, we describe our simulation model for performance evaluation. The model is based on the system architecture described in Section 3 and is implemented by CSIM19 [38]. The model includes a single server and a number of clients (c). It is an open system in which all clients continually issue requests. The inter-arrival time, namely, the difference between arrival time of two consecutive requests, is exponentially distributed. The mean is set to be a constant divided by f, which is actually a scaling factor of the request arrival

rate. In our simulations, the mean request inter-arrival time for each client is set to be $\frac{10}{f}$ time units, and hence the overall mean request inter-arrival time of the system is $\frac{10}{c*f}$ time units, where c is the number of clients.

Algorithm 1: Profit Versus Cost (PVC)

```
1  /* all parameters are assumed to be zero as default value before reassignment */
2  begin Part 1: Arrival of a new request
3      /* add the new request Q_j into PendingQ, update profit, ProfitQ and SlackQ */
4      PendingQ ← PendingQ ∪ {Q_j};
5      for d_i ∈ Q_j do
6          profit(d_i|Q_j) ← 1/n_j;
7          profit(d_i) ← profit(d_i) + profit(d_i|Q_j);
8          for Q ⊇ {d_i} do
9              avgprofit(Q) ← avgprofit(Q) + profit(d_i|Q_j)/k;
10             Update the position of Q in ProfitQ in descending order of avgprofit;
11     Insert Q_j to ProfitQ in descending order of avgprofit;
12     Insert Q_j to SlackQ in ascending order of deadline Dl_j;
13 end
14 begin Part 2: Item Selection and Broadcast
15     /* remove infeasible requests */
16     for Q_i ∈ PendingQ do
17         s_i ← (Dl_i − clock)/b;
18         if s_i < k_i then
19             PendingQ ← PendingQ − {Q_i};
20     /* calculations of scheduling weights PVC */
21     T ← 0;
22     S_min ← SlackQ.head.s;
23     for Q+ ∈ ProfitQ, where (avgprofit(Q+) > T * S_min) do
24         Q ← Q+;
25         Q.isExaminedFlag ← 1;
26         S_min ← min_{Q− ∈ SlackQ} {s(Q−)|Q−.isExaminedFlag == 0};
27         for Q_j ≠ Q do
28             E(Q_j|Q) ← (k + k_j > s_j)? k/k_j : 0;
29             for d_i ∈ Q_j do
30                 if d_i ∉ Q then
31                     E(d_i|Q) ← E(d_i|Q) + E(Q_j|Q);
32         C(Q) ← max_{d_i ∉ Q} {E(d_i|Q)};
33         PVC(Q) ← (avgprofit(Q) − C(Q))/s(Q);
34         if PVC(Q) > T then
35             T ← PVC(Q);
36             selectedRequest Q_s ← Q;
37     P_max ← max_{Q+ ∈ ProfitQ} {avgprofit(Q+)|Q+.isExaminedFlag == 0};
38     for Q− ∈ SlackQ, where (s− < P_max/T) do
39         Q ← Q−;
40         Q.isExaminedFlag ← 1;
41         P_max ← max_{Q+ ∈ ProfitQ} {avgprofit(Q+)|Q+.isExaminedFlag == 0};
42         Repeat lines 27–36;
43     /* item selection and scheduling decision */
       selectedItem d_s ← d_i, where d_i ∈ Q_s and profit(d_i) == max_{d_i ∈ Q_s} {profit(d_i)};
44
45     Broadcast the selected data item d_s;
46 end
47 begin Part 3: Updating of the PendingQ after broadcasting
48     /* remove item d_s from the requests requiring it */
49     Δ ← 0;
50     for Q_j ∈ PendingQ, where d_s ∈ Q_j do
51         if Q_j retrieves d_s from broadcasting then
52             Δ ← Δ + profit(d_s|Q_j);
53             profit(d_s) ← profit(d_s) − profit(d_s|Q_j);
54     for Q_j ∈ PendingQ, where d_s ∈ Q_j do
55         avgprofit(Q_j) ← (k_j * avgprofit(Q_j) − Δ)/(k_j − 1);
56         Q_j ← Q_j − {d_s};
57         if k_j − − == 0 then
58             PendingQ ← PendingQ − {Q_j};
59         Update the position of Q_j in ProfitQ in descending order of avgprofit;
60 end
```

The number of items in a request follows the Uniform distribution in $[\overline{n}-1, \overline{n}+1]$ with expectation $\overline{n}$ when $\overline{n}>1$. When $\overline{n}=1$, all requests are single-item requests. Deadline Dl_i of a request Q_i is set according to the following formula, where n_i is the initial number of items in the request and *uniform* produces a random number selected uniformly from the range of request deadline scaling factor *LAX_MIN*~*LAX_MAX*.

$$Dl_i = AT_i + (1 + \mathit{uniform}(\mathit{LAX\_MIN}, \mathit{LAX\_MAX})) \cdot n_i$$

Note that each item has a unit size. The data access pattern is shaped with the Zipf distribution [39] according to widely accepted research on human behavior patterns, with skewness parameter θ, where $0 \le \theta \le 1$. The smaller the value of skewness parameter is, the less different are item access frequencies among all items in the database. When $\theta = 0$, the distribution becomes the Uniform distribution, while the Zipf distribution becomes strict if θ increases to 1. In response to requests in the pending queue, the server continually retrieves data items from the database of size N for broadcasting while clients listen to the single channel to retrieve their data items of interest. The parameters and their default setting can be found in Table 4.

4.3.3. Performance metrics

We now introduce the following performance metrics.

$$①\,Request\ \ Deadline\ \ Miss\ \ Ratio = \frac{TotalMissedNum}{TotalNum}$$

$$②\,Ratio\ \ of\ \ Ineffective\ \ Service\ \ Over\ \ Effective\ \ Service = \frac{TotalIneffectiveNum}{TotalSatisfiedNum}$$

As the primary performance metric, request deadline miss ratio shown in Metric ① is the percentage of missing requests in all requests during the time span of a whole simulation. Correspondingly, in Metric ①, *TotalMissedNum* represents the total number of deadline-missed requests, and *TotalNum* represents the total number of submitted requests. A low request deadline miss ratio reflects a better real-time performance in terms of satisfying requests with time constraints.

Ratio of ineffective service over effective service shown in ② is another auxiliary performance metric to interpret variations of request deadline miss ratios of different algorithms in simulations. Ineffective service means the service for the requests that are served partially and whose deadlines are missed finally. Effective service means the service for the requests satisfied finally. In Metric ②, *TotalIneffectiveNum* is the number of requests that are served partially and whose deadlines are missed finally, *TotalSatisfiedNum* is the number of requests satisfied finally. Note that $TotalSatisfiedNum = TotalNum - TotalMissedNum$. A lower ratio of ineffective service over effective service means fewer instances of ineffective service provided for satisfying requests with time constraints.

4.3.4. Result analysis

For performance evaluation, we adopt the primary performance metric in real-time systems, namely, the request deadline miss ratio, which reflects the capability of the system to meet request deadlines. The prime objective of scheduling algorithms is to improve real-time performance of the system by minimizing the request deadline miss ratio. The ratio of ineffective service over effective service is another auxiliary performance metric that reflects the number of instances of ineffective service being provided by different algorithms. Note that the simulation results were obtained until a confidence interval of 0.95 with half-widths of less than 5% about the mean was achieved.

Before discussing the ratio of ineffective service over effective service, we focus on request deadline miss ratio in Figs. 3–8. After that, simulation results on the ratio of ineffective service over effective service are given in Fig. 9.

Fig. 3 shows simulation results for different request arrival rates scaling factor f. On the whole, we can see that all our profit-based algorithms outperform other algorithms in their respective groups. The advantage of Algorithm PVC over other algorithms increases with increase of f.

Fig. 4 shows simulation results for different skewness parameters of Zipf distribution θ. As a high skewness parameter is favorable for satisfaction of requests that include hot items, request deadline miss ratio decreases when θ increases for each algorithm. Algorithm PVC has larger advantage over other algorithms when θ is smaller. In addition, Algorithm FCFS outperforms other algorithms of Group B when θ increases to 1.

Fig. 5 shows simulation results for different average numbers of items in a request under fixed default workloads. On the whole, PVC always outperforms other algorithms.

Table 4
Parameters and default setting.

Parameter	Default	Range
f	40	10–80
θ	0.5	0.0–1.0
c	30	–
M	6×10^4	–
$\overline{n}$	4	1–8
n_i	3–5	1–9
N	200	200–3200
b	1.0	0.7–1.3
LAX_MIN	15	5–25
LAX_MAX	25	15–35

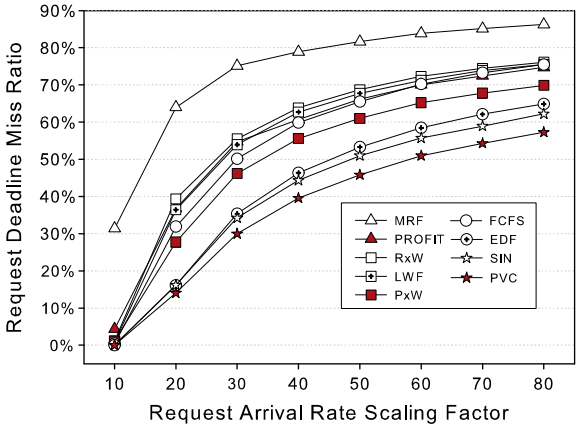

Fig. 3. Request deadline miss ratio under different request arrival rates.

For Group A and Group B, we can see that PROFIT is always better than MRF and PxW is always better than RxW. This result reflects that the concept profit is more effective than item access frequency under the same condition, namely with or without the factor of waiting time. However, FCFS changes to be better than RxW and PxW when average number of items in a request increases to more than 4.

For Group C, SIN changes to be worse than EDF when average number of items in requests increases to more than 4. This reflects that under the condition of same deadlines of requests, EDF is more adaptable than SIN to scheduling requests requiring more items. In other words, EDF keeps good performance even when average number of items in requests increases. In comparison to EDF, PVC keeps better performance with the increasing average number of items in requests. It shows that PVC is more adaptable than EDF to scheduling requests requiring more items when deadlines are the same for all requests. A direct reason for this result is that PVC prefers to satisfy requests closer to completion.

Fig. 6 shows simulation results for different data transmission rates, i.e. different bandwidths. Generally, the proposed profit-based algorithms always outperform other algorithms in their respective groups. In addition, performance curves of all algorithms are descending with increase of bandwidth. This indicates that increasing data transmission rate improves the real-time performance in terms of request deadline miss ratio.

Fig. 7 shows simulation results for different ranges of request deadline scaling factor (*LAX_MIN~LAX_MAX*). When the mean value of the range of request deadline scaling factor increases, request deadline miss ratio decreases. Algorithm PVC has larger advantage over other algorithms when the mean value of the range of request deadline scaling factor is smaller.

Fig. 8 shows the simulation results for different numbers of data items in the database. With the increasing number of data items in the database, profit-based algorithms have increasing performance superiority in comparison with non-profit-based algorithms in their respective groups. PROFIT shows great performance robustness when the number of data items in the database is larger than 400. PROFIT is even better than EDF, SIN of Group C when the number of data items in the database increases to 3200. PxW of Group B changes to be better than SIN of Group C when the number of data items in the database increases to more than 800. Overall, profit-based algorithms have better scalability to large database than non-profit-based algorithms.

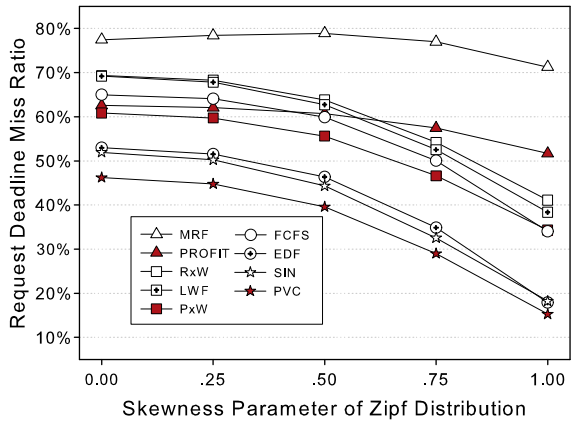

Fig. 4. Request deadline miss ratio under different skewness parameters of Zipf distribution.

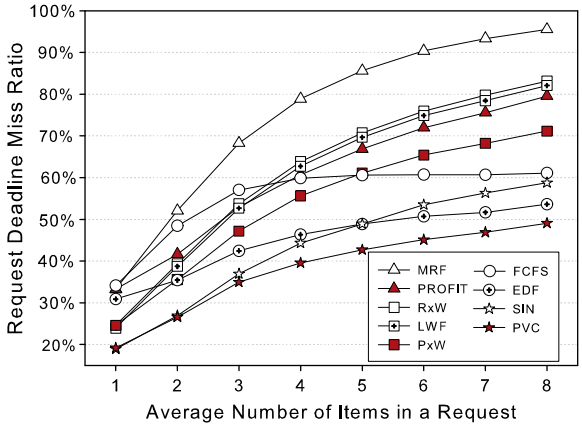

Fig. 5. Request deadline miss ratio under different average numbers of items in a request (workloads, i.e. $c{\cdot}f{\cdot}\bar{n}$, are fixed to default).

Fig. 9 shows the ratio of ineffective service over effective service under different request arrival rates. In general, the ratio of ineffective service over effective service of all algorithms increases linearly with increase of request arrival rate scaling factor. Specifically, profit-based algorithms outperform other algorithms in their respective groups. This indicates that fewer excess requests have been served partially and their deadlines are missed finally in case of profit-based algorithm than other algorithms in their respective groups. In contrast with Fig. 3, Fig. 9 also indicates that resources saved by reducing ineffective service improves the real-time performance in terms of request deadline miss ratio.

To conclude, Algorithm PVC outperforms other algorithms in the three groups for different request arrival rates, different skewness parameters of data access pattern, different average numbers of items in a request, different bandwidths, different deadline ranges and different database sizes. In addition, our simulation results show that EDF and FCFS have preferable performance with respect to low complexity algorithms for lighter workloads, larger skewness of data access pattern and larger average number of items in a request. Our simulation results also show that PROFIT and PxW are preferable algorithms in their respective groups.

5. Multi-channel scheduling

5.1. Hardness

The hardness of multi-channel scheduling cannot be directly derived from the proof for single channel environment. As shown in the following theorem, multi-channel scheduling is also NP-hard.

Theorem 2. *Broadcast scheduling of multi-item requests with deadline constraint in multi-channel environment is NP-hard.*

Proof. Suppose there are K channels that have the same bandwidth 1. We construct a polynomial time reduction from given Instance 1 $\langle m, \{(n_i, AT_i, Dl_i)|i=1,\ldots,m\}, C\rangle$ of single channel scheduling to an instance of multi-channel scheduling, called Instance 2. The problem is to ask if this instance has a schedule that can satisfy at least $C+m\cdot(K-1)\cdot\Delta t$ requests, where Δt is the scheduling period for all requests in Instance 1.

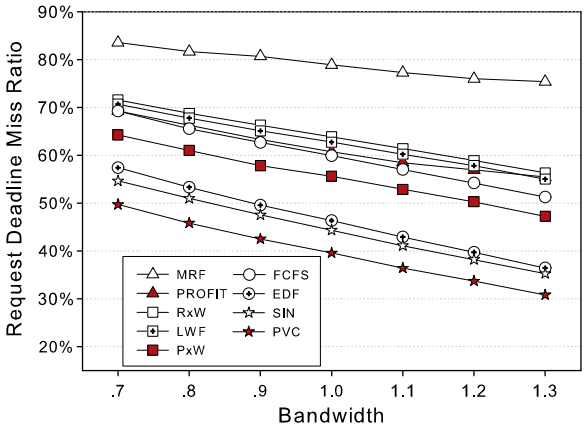

Fig. 6. Request deadline miss ratio under different bandwidths.

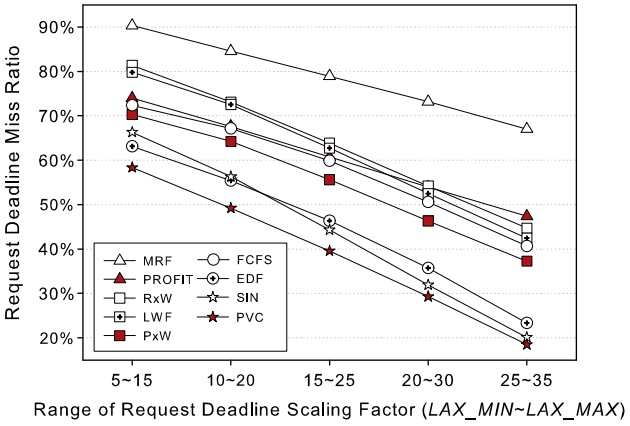

Fig. 7. Request deadline miss ratio under different ranges of request deadline scaling factor.

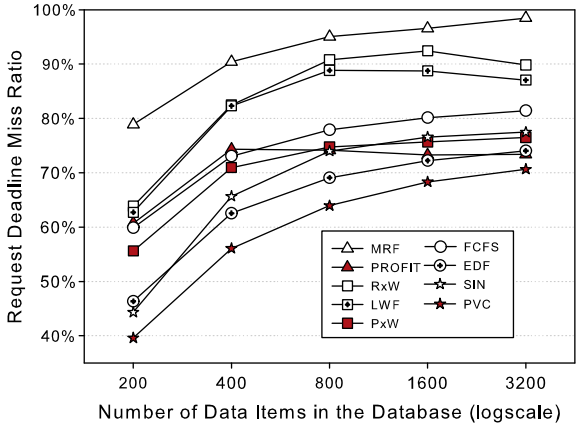

Fig. 8. Request deadline miss ratio under different numbers of data items in the database.

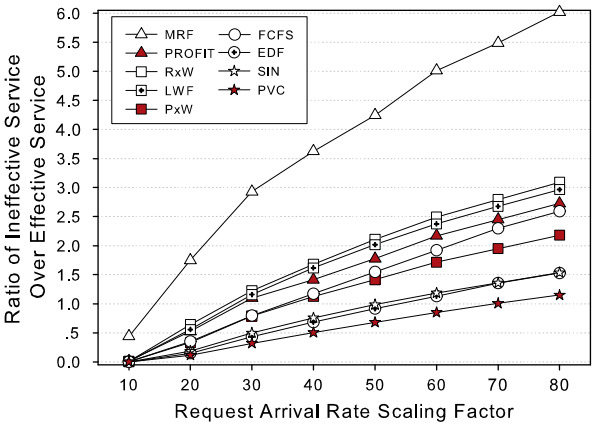

Fig. 9. Ratio of ineffective service over effective service under different request arrival rates.

For Instance 1, we map all m requests directly to Instance 2. We call these directly mapped requests ordinary requests. Additionally, we add $(K-1)$ groups of single-item requests. For group i, m requests requiring a unique same item X_i are released for every time slot within the scheduling period Δt. That is, there are totally $|\Delta R| = m \cdot (K-1) \cdot \Delta t$ special requests added to Instance 2.

A simple example of the above transformation is shown in Fig. 10. This reduction process can be completed in polynomial time. Next, we prove the correctness of this reduction.

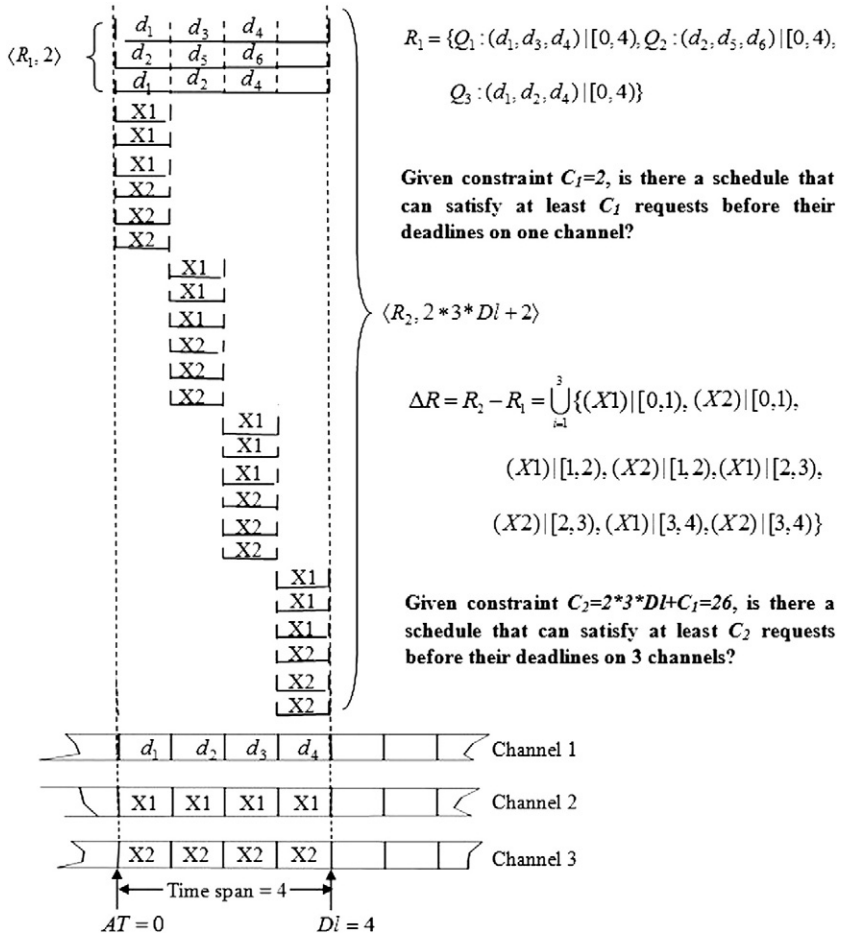

Fig. 10. An example of reduction process on Instance 2 for 3 channels.

Suppose Instance 1 has a schedule that satisfies C requests within the scheduling period Δt. Then, for each time slot, the other $(K-1)$ channels can satisfy $(K-1)$ groups of special requests, i.e. $m\cdot(K-1)$ requests. So for scheduling period Δt, $(K-1)$ channels can satisfy all $m\cdot(K-1)\cdot\Delta t$ special requests. Plus the C ordinary requests that the remaining one channel can satisfy, there is a schedule that can satisfy $C+m\cdot(K-1)\cdot\Delta t$ requests.

Conversely, suppose the constructed Instance 2 can satisfy $C+m\cdot(K-1)\cdot\Delta t$ requests within the scheduling period Δt. Then, as indicated before, the optimal solution should make $(K-1)$ channels occupied by the different $(K-1)$ data items for $m\cdot(K-1)$ special requests for each time slot. Otherwise, the deadlines of at least m special requests for item X_i will be missed for some time slot. Then the schedule can satisfy at most $m\cdot(K-1)\cdot\Delta t-m$ special requests. As the schedule can satisfy at most m ordinary requests, the total number of satisfied requests will be less than or equal to $m\cdot(K-1)\cdot\Delta t$. This contradicts the premise. Therefore, the $(K-1)$ channels should be utilized to satisfy all $|\Delta R|=m\cdot(K-1)\cdot\Delta t$ special requests. So the remaining one channel can satisfy $C+m\cdot(K-1)\cdot\Delta t-|\Delta R|=C$ requests, i.e. Instance 1 can satisfy C requests within the scheduling period Δt.

As shown in Theorem 1, single channel scheduling is NP-hard. Theorem 2 is proved. □

5.2. Proposed algorithm SSA (Single Slot Allocation)

In the system model of Section 3, we know that every un-satisfied request can obtain one item from only one channel during one time slot. That is, for the current time slot, requests/clients are either partitioned and allocated to one of the given multiple channels for receiving required item or put aside for the next time slot. Therefore, after a request/client is allocated to one channel for receiving one required item, it cannot simultaneously receive another required item on another channel. We call this phenomenon conflict of inter-channel allocation. An effective algorithm for multi-channel scheduling should reduce this conflict [29].

For example, suppose there are four newly arrived requests at time 0[2]:

Example 3. Q_1: (d_1, d_2)||[0, 2), Q_2: (d_2, d_3)||[0, 2), Q_3: (d_1, d_4)||[0, 2) *and* Q_4: (d_3, d_5)||[0, 2).

[2] Recall that a request Q, which arrives at time AT with deadline Dl and requires an item set S, is represented as S||[AT,Dl).

As items d_1, d_2 and d_3 have the same popularity, a possible schedule will be $d_1 \rightarrow d_2$ on channel 1 and $d_2 \rightarrow d_3$ on channel 2. This schedule will satisfy 2 requests. The problem is that the client submitting Q_1 can only listen to one channel at one time, i.e. either channel 1 or channel 2. So if it listens to channel 1, item d_2 on channel 2 will only serve request Q_2. However, if we move Q_1 in the pending queue to a group for channel 1, we can easily find that item d_3 becomes the most popular item to be scheduled on channel 2. Then a better schedule will be $d_1 \rightarrow d_4$ on channel 1 and $d_3 \rightarrow d_2$ on channel 2 which can satisfy 3 requests. It is not hard to see that grouping is a feasible method to reduce the conflict of inter-channel allocation.

Considering the need to keep conflict of inter-channel allocation low, a direct way to improve real-time performance is to serve as many requests as possible during one time slot. However, it does not mean that all requests ever getting served before are satisfied finally. Therefore, requests selected for allocation to multiple channels should be representative of all requests. We should not only serve as many requests as possible but also satisfy as many requests ever getting served before as possible.

Based on the above observation, we propose a single slot allocation Algorithm SSA. The algorithm can be partitioned into two parts, scheduling part and allocation part.

The scheduling part is to serve as many requests as possible during one time slot. We utilize *PVC* values of requests to measure the value of "representative" (representativeness) for all requests. Within a representative request, we select and broadcast the item with the largest value of absolute profit of unserved items, which is called the "characteristic item" of the request. The above task is completed by scheduling algorithm PVC.

The allocation part is to reduce the conflict of inter-channel allocation and to satisfy as many requests ever served before as possible. The basic process of the allocation part is to allocate K different items on K available channels, if possible. The process is implemented by at most K iterative partitioning as follows. Originally, all requests are moved from a set *PendingQ* to a set *WaitingQ* for scheduling. At the beginning of each iterative partitioning, a most representative request from *WaitingQ* with the largest *PVC* value and its characteristic item are selected to be allocated to an available channel. The characteristic item is, therefore, to be broadcast on one available channel. Next, all requests from *WaitingQ* requiring the characteristic item are moved to a set *ServiceQ*. That is, the clients and their submitted requests requiring the characteristic item form a group on the available channel waiting for service. In addition, the remaining requests of *WaitingQ* requiring any other item (in contrast to the characteristic item) of the representative request are moved to a set *ShelvedQ*. Requests in *ShelvedQ* are put aside until they are moved to *WaitingQ* when *WaitingQ* is empty and an available channel exists. So after each iterative partitioning, some new clients and their submitted requests are grouped together on a new available channel waiting for service. The above tasks are completed after at most K iterative partitioning with grouping process by utilizing three sets, namely *WaitingQ*, *ServiceQ* and *ShelvedQ*. Details of the algorithm are shown in Algorithm 2.

Algorithm 2: Single Slot Allocation (SSA)

```
1  begin
2      Remove infeasible requests in PendingQ;
3      /* initialization of WaitingQ, ServiceQ and ShelvedQ, item set D_b to be broadcast and item set D_s to
          be shelved                                                                                    */
4      WaitingQ ←− PendingQ ;
5      ShelvedQ ←− ServiceQ ←− PendingQ ←− ∅ ;
6      D_s ←− D_b ←− ∅;
7      /* scheduling, allocation and broadcast                                                          */
8      repeat
9          if WaitingQ == ∅ then
10             Swap (WaitingQ, ShelvedQ);
11             D_s ←− ∅ ;
12         Find a request Q_0 from WaitingQ as a representative request that has the largest value of PVC;
13         Select an item d_s from Q_0 that has the largest value of profit out of the unserved items of Q_0;
14         D_b ←− D_b ∪ {d_s};
15         D_s ←− D_s ∪ Q_0 − D_b;
16         ServiceQ ←− ServiceQ ∪ {Q_0};
17         WaitingQ ←− WaitingQ − {Q_0};
18         for each Q_i ∈ WaitingQ such that d_s ∈ Q_i do
19             ServiceQ ←− ServiceQ ∪ {Q_i};
20             WaitingQ ←− WaitingQ − {Q_i};
21         for each Q_i ∈ WaitingQ such that the client that submits Q_i is in service for Q_0 do
22             PendingQ ←− PendingQ ∪ {Q_i};
23             WaitingQ ←− WaitingQ − {Q_i};
24         for each Q_j ∈ WaitingQ such that D_s ∩ Q_j ≠ ∅ do
25             ShelvedQ ←− ShelvedQ ∪ {Q_j};
26             WaitingQ ←− WaitingQ − {Q_j};
27         Broadcast item d_s on an available channel;
28     until no available channel or WaitingQ ∪ ShelvedQ == ∅ ;
29     PendingQ ←− PendingQ ∪ WaitingQ ∪ ServiceQ ∪ ShelvedQ;
30 end
```

For the time complexity of the algorithm, we argue that the algorithm can be completed in time $O(K^2 \cdot n \cdot m + K \cdot m^2)$, where n is the average number of items for all requests. Specifically, time is mainly consumed in the iterative process for lines 8–28 (the remaining part of the algorithm can be completed in time $O(m)$). Line 12 can be completed in time $O(m^2 + m) = O(m^2)$ as the scheduling part of the algorithm is executed in this line. Lines 18–20 can be completed in time $O(n \cdot m)$. Lines 21–23 can be completed in time $O(m)$. Lines 24–26 can be completed in time $O(K \cdot n \cdot m)$. Other lines of the iterative process can be completed in time $O(n)$. As a result, with at most K times of iterative process, we get the time complexity $O(K^2 \cdot n \cdot m + K \cdot m^2)$.

To combine an index generation method with the scheduling in multi-channel environment, we can interleave index segments with data segments on each channel.

Specifically, series of buckets make up the broadcast sequences on the channels [32]. A bucket spanning multiple channels contains one index segment (IS) and one data segment (DS) (Fig. 11). The index segment consists of index items linked to all data items in the data segment of the same bucket as well as the broadcast time of the next bucket. Each index item contains the channel identifier and broadcast time of the linked data item. Note that the index items are duplicated on each channel such that clients listening to any channel can retrieve full index information from the index segment. By scanning the index segment, clients can switch to the corresponding channel for retrieving the desired data item. Otherwise, if no desired item is found in the index segment, clients can wait until the next index segment arrives. To ensure enough time for clients to switch channel, there is a preamble type field (P) at the beginning of the data segment for channel synchronization [7]. The data segment consists of the payload type field of data items, namely the data items themselves, and some auxiliary type fields, such as the preamble described above and forward error correction (FEC) code for possible data recovery [7]. The number of data items stored in a bucket is called the degree of bucket, which is a multiple of the number of channels. Fig. 11 shows a 3-channel example and the degree of bucket is 3. For each bucket, one channel stores one data item in its data segment and the index segment consists of index items linked to all the three data items in the same bucket.

As there may be more than one data item requested by a client, it is possible that different data items requested by the same client are broadcast on different channels. To resolve this issue, the index items can be stored in descending order of absolute profit of the corresponding data items. Therefore, the earliest index item scanned by a client will link to the desired data item of the client. For example, we assume $profit(d_3) > profit(d_2) > profit(d_1)$ in Fig. 11 and the index items are stored in order of I_3, I_2 and I_1. Therefore, a client requesting both d_3 and d_2 will be directed to d_3 by the sorted index segment.

5.3. Performance evaluation

5.3.1. Experimental preparation and setup

As simulations of scheduling of real-time multi-item requests were performed in multi-channel environment, we introduce two traditional algorithms, SIN and EDF, for comparison and modify them to adapt to the multi-channel environment. The modifications are as follows.

Being a scheduling algorithm based on item level scheduling weights, SIN is changed to select K different items with the minimum values of SIN and to allocate the selected items to all channels for one time slot. Being a scheduling algorithm based on request level scheduling weights, EDF is changed to select K requests with the earliest deadlines and then to choose within each request one unserved item to broadcast on an available channel for one time slot. In addition, the scheduler of EDF should try its best to ensure that the selected items are different from each other, if possible.

So the related traditional algorithms are a little different from the original ones. In the following, while not confusing the two algorithms and their respective original versions, we directly use SIN and EDF to represent the corresponding modified versions for multi-channel scheduling.

To fairly compare SSA with the modified traditional algorithms in multi-channel environment, we set up a simulation model for multi-channel scheduling based on the existing model for single channel scheduling shown in Section 4.3.2. The parameters are set as follows. The number of channels is K. The bandwidth of each channel is $b_i = b/K$ where $b = 1$ in the single channel case. That is, the total bandwidth of K channels is the same as the single channel in single channel environment. As a default, $K = 3$. The Zipf distribution parameter θ is the same as the default value of single channel scheduling. In addition, all requests in multi-

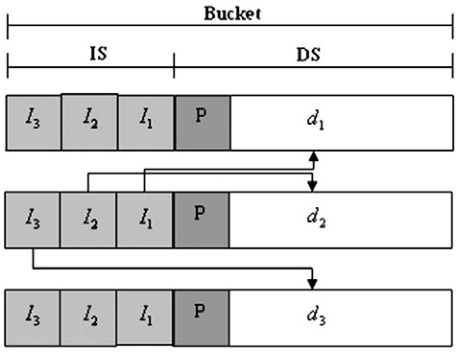

Fig. 11. Multi-channel index structure.

channel environment are the same as the ones produced under default settings in single channel environment, including deadlines. Note that as parameters *LAX_MIN* and *LAX_MAX* of uniform distribution for generating deadlines of requests are large enough in the default setting (15 and 25), all feasible requests in single channel environment are also feasible in multi-channel environment. The only difference is that urgent requests become more urgent when a smaller bandwidth is available for each client. For a general view of the model, please see Section 3 and Fig. 1.

5.3.2. Performance metrics

To evaluate and analyze the simulation results, we use the following performance metrics.

$$①Request\ \ Deadline\ \ Miss\ \ Ratio = \frac{TotalMissedNum}{TotalNum}$$

$$②Ratio\ \ of\ \ Ineffective\ \ Service\ \ Over\ \ Effective\ \ Service = \frac{TotalIneffectiveNum}{TotalSatisfiedNum}$$

$$③Service\ \ Efficiency = \frac{\sum_{i=1}^{K} ServedNum_i}{\sum_{i=1}^{K} PendingNum_i}$$

For the first two performance metrics, please refer to Section 4.3.3. In the following, we discuss Metric ③.

Service efficiency shown in Metric ③ is an auxiliary performance metric that reflects the level of conflict of inter-channel allocation. For a given time slot, as a client together with its submitted requests can retrieve only one required item from one channel at a time, other required items on other channels, if any, are missed. In Metric ③, for one time slot, $ServedNum_i$ is the number of requests getting served on channel *i*, $PendingNum_i$ is the total number of requests requiring the broadcasted item on channel *i*. So the metric is actually the percentage of accumulated sum of served items ($ServedNum_i$ on channel *i*) for all requests getting served at a time slot to the accumulated sum of item access frequency ($PendingNum_i$ on channel *i*) for all broadcasted items on all channels during the time slot. For a whole simulation, the metric implies the average value of the percentage for all time slots. A lower service efficiency means on average more missed items for requests getting served during the time slot. In other words, a lower service efficiency means high level of conflict of inter-channel allocation. Note that this metric is derived from a similar metric in a previous work of K. Liu et al. [29].

5.3.3. Result analysis

According to the performance metrics introduced in Section 5.3.2, we perform corresponding simulations to compare SSA with two classical algorithms, EDF and SIN, as outlined in Section 5.3.1. Note that the simulation results were obtained until a confidence interval of 0.95 with half-widths of less than 5% about the mean was achieved.

Fig. 12 shows variations of request deadline miss ratio of different algorithms with increasing number of channels. Generally, SSA performs better than SIN while SIN is better than EDF. Specifically, when the number of channels is 5, the difference between SSA and SIN is about 20%. From the perspective of performance variation with increasing number of channels, performance of SSA is more steady than SIN and EDF, which means request deadline miss ratio of SSA increases more slowly than SIN and EDF, with increasing number of channels. Note that in spite of performance of EDF being the poorest, as shown in the figure, its performance changes more slowly than SIN when the number of channels increases to more than 2.

Fig. 13 shows the ratio of ineffective service over effective service under different numbers of channels. The increasing performance advantage of SSA over EDF and SIN with the increasing number of channels indicates that fewer requests have been served

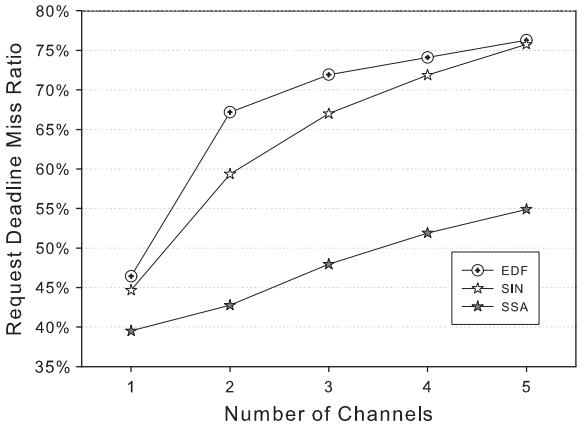

Fig. 12. Request deadline miss ratio under different numbers of channels.

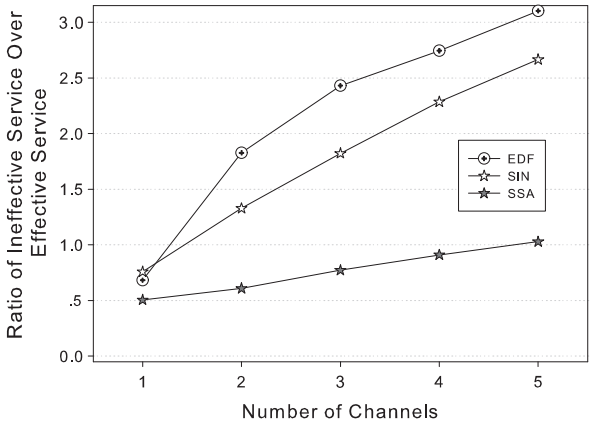

Fig. 13. Ratio of ineffective service over effective service under different numbers of channels.

partially and their deadlines are missed finally for SSA than for EDF and SIN. In other words, SSA provides the least instances of ineffective service in the aspect of satisfying requests with time constraints.

Fig. 14 shows service efficiency under different numbers of channels. SSA performs the best, EDF takes the second place, and SIN has the worst performance. Based on discussions on performance metrics in Section 5.3.2, SSA and EDF have lower levels of conflict of inter-channel allocation than SIN.

Fig. 15 shows variations of request deadline miss ratio of different algorithms with increasing average number of items in a request. Generally, SSA always wins a larger performance advantage over SIN than SIN over EDF.

Fig. 16 shows variations of request deadline miss ratio of different algorithms with increasing number of items in the database. The great performance advantage of SSA over EDF and SIN indicates that SSA has good scalability to large databases.

To conclude, SSA has the lowest level of conflict of inter-channel allocation and provides the least instances of ineffective service. Hence, SSA has the best performance in terms of request deadline miss ratio under different numbers of channels, different average numbers of items in a request and different database sizes.

6. Conclusion

In this paper, we first prove NP-hardness of the problem of optimization of minimizing request deadline miss ratio for both uniform single channel case and uniform multi-channel case. Then we propose two scheduling algorithms for single channel scheduling and multi-channel scheduling.

For single channel scheduling of multi-item requests, we introduce a new concept called absolute profit of an item to effectively denote its contribution to the system's throughput. In addition, to calculate the potential opportunity cost, we classify the relationship between two requests into unidirectional compatible class and unidirectional exclusive class. Based on the classifications, we quantify opportunity cost. After that, we propose three profit-based scheduling algorithms called PROFIT,

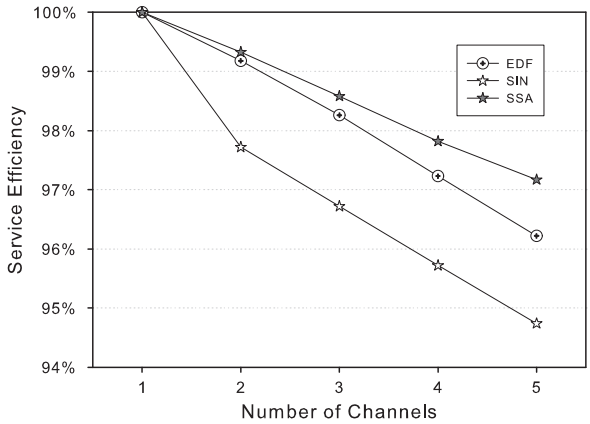

Fig. 14. Service efficiency under different numbers of channels.

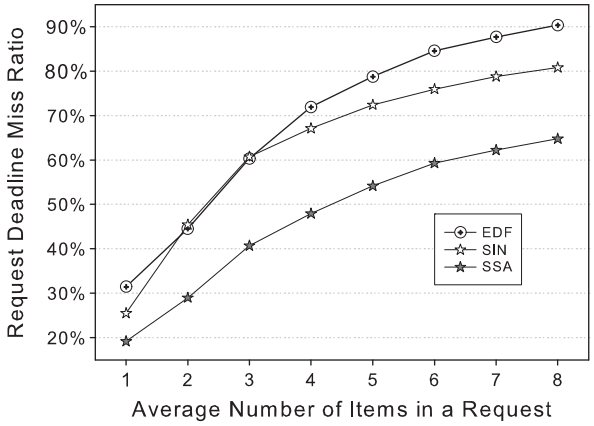

Fig. 15. Request deadline miss ratio under different average numbers of items in a request (workloads, i.e. $c{\cdot}f{\cdot}\bar{n}$, are fixed to default).

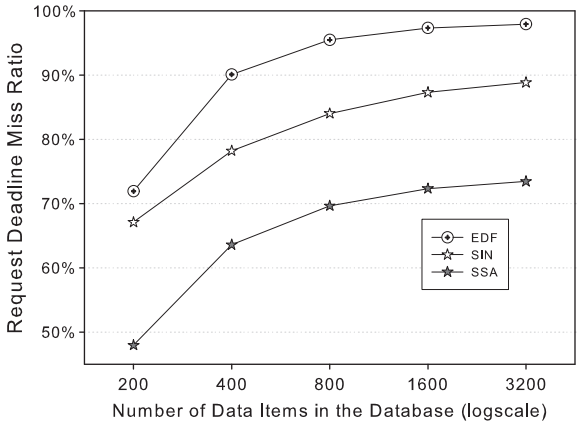

Fig. 16. Request deadline miss ratio under different numbers of data items in the database.

PXW and PVC. The simulations show that profit-based algorithms, especially PVC, outperform other traditional algorithms for scheduling multi-item requests in single channel environment.

For multi-channel scheduling of multi-item requests, based on the scheduling result of single channel scheduling algorithm PVC for pending requests, we further propose Algorithm SSA to allocate selected items of scheduled requests to available channels. The allocations mainly use three sets, namely, *WaitingQ*, *ServiceQ* and *ShelvedQ*, to perform an iterative partitioning process for the pending requests, with grouping process. After each iterative partitioning, an item from a scheduled request can be selected and allocated to an available channel based on Algorithm PVC. Simulations verify the great effectiveness of SSA.

In the future, we propose to investigate if there exists any Polynomial-Time Approximation Scheme (PTAS) with good competitive ratio for some special case. Additionally, we plan to further study two-dimensional scheduling for multi-channel multi-slot environment. Meanwhile, we will improve the Quality of Service (QoS) under the condition of balancing the system's real-time performance.

Acknowledgment

The work described in this paper was partially supported by grants from Natural Science Foundation of China (Grant No. 61073110), the Key Program of National Natural Science Foundation of China (Grant No. 60933013), and Research Fund for the Doctoral Program of Higher Education of China (20093402110017), and City University of Hong Kong (Project No. 7008042).

References

[1] H. Fang, T. Tao, C. Zhai, Diagnostic evaluation of information retrieval models, ACM Transactions on Information Systems (TOIS) 29 (2) (2011) 7.
[2] J. Chang, T. Erlebach, R. Gailis, S. Khuller, Broadcast scheduling: algorithms and complexity, Proceedings of the nineteenth annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, Philadelphia PA, USA, 2008, pp. 473–482.
[3] S. Yi, H. Shin, A hybrid scheduling scheme for data broadcast over a single channel in mobile environments, Journal of Intelligent Manufacturing (2010) 1–11.

[4] S. Acharya, R. Alonso, M. Franklin, S. Zdonik, Broadcast disks: data management for asymmetric communication environments, International Conference on Management of Data: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, 1995.
[5] Y. Chang, C. Yang, A complementary approach to data broadcasting in mobile information systems, Data & Knowledge Engineering 40 (2) (2002) 181–194.
[6] W. Yee, S. Navathe, E. Omiecinski, C. Jermaine, Efficient data allocation over multiple channels at broadcast servers, IEEE Transactions on Computers 51 (10) (2002) 1231–1236.
[7] S. Lo, A. Chen, Efficient index and data allocation for wireless broadcast services, Data & Knowledge Engineering 60 (1) (2007) 235–255.
[8] S. Anticaglia, F. Barsi, A. Bertossi, L. Iamele, M. Pinotti, Efficient heuristics for data broadcasting on multiple channels, Wireless Networks 14 (2) (2008) 219–231.
[9] P. Yu, W. Sun, Y. Qin, Z. Zhang, B. Shi, A data partition based near optimal scheduling algorithm for wireless multi-channel data broadcast, Lecture Notes in Computer Science 4947 (2008) 188.
[10] I. Michalarias, A. Omelchenko, H. Lenz, Fclos: a client-server architecture for mobile OLAP, Data & Knowledge Engineering 68 (2) (2009) 192–220.
[11] D. Aksoy, M. Franklin, R×W: a scheduling approach for large-scale on-demand data broadcast, IEEE/ACM Transactions on Networking (TON) 7 (6) (1999) 846–860.
[12] J. Wong, M. Ammar, Analysis of broadcast delivery in a videotex system, IEEE Transactions on Computers 100 (34) (1985) 863–866.
[13] J. Wong, Broadcast delivery, Proceedings of the IEEE 76 (12) (1988) 1566–1577.
[14] W. Ni, Q. Fang, S. Vrbsky, A Lazy Data Request Approach for On-Demand Data Broadcasting, Proceedings of the 23rd International Conference on Distributed Computing Systems, IEEE Computer Society, Washington DC, USA, 2003.
[15] D. Aksoy, M. Leung, Pull vs push: a quantitative comparison for data broadcast, IEEE Global Telecommunications Conference, GLOBECOM'04, vol. 3, 2004.
[16] S. Kang, Wireless data broadcast scheduling with utility metric based on soft deadline, IEICE Transactions on Communications 94 (5) (2011) 1424–1431.
[17] N. Saxena, C. Pinotti, S. Das, A probabilistic push–pull hybrid scheduling algorithm for asymmetric wireless environment, IEEE Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004, 2004, pp. 5–9.
[18] J. Xu, X. Tang, W. Lee, Time-critical on-demand data broadcast: algorithms, analysis, and performance evaluation, IEEE Transactions on Parallel and Distributed Systems 17 (1) (2006) 3–14.
[19] C. Liu, J. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, Journal of the Association for Computing Machinery 20 (1) (1973) 46–61.
[20] V. Lee, X. Wu, J. Ng, Scheduling real-time requests in on-demand data broadcast environments, Real-Time Systems 34 (2) (2006) 83–99.
[21] R. Dewri, I. Ray, et al., Utility driven optimization of real time data broadcast schedules, Applied Soft Computing (2011) to appear.
[22] W. Chen, P. Chundi, Extracting hot spots of topics from time-stamped documents, Data & Knowledge Engineering 70 (7) (2011) 642–660.
[23] D. Yang, M. Chen, Data broadcast with adaptive network coding in heterogeneous wireless networks, IEEE Transactions on Mobile Computing (2009) 109–125.
[24] Y. Chung, M. Kim, QEM: a scheduling method for wireless broadcast data, Proceedings of the Sixth International Conference on Database Systems for Advanced Applications, IEEE Computer Society, 1999, pp. 135–142.
[25] W. Sun, Z. Zhang, P. Yu, Y. Qin, Skewed wireless broadcast scheduling for multi-item queries, IEEE International Conference on Wireless Communications, Networking and Mobile Computing, 2007, WiCom2007, 2007, pp. 1865–1868.
[26] J. Chen, V. Lee, K. Liu, On the performance of real-time multi-item request scheduling in data broadcast environments, Journal of Systems and Software 83 (8) (2010) 1337–1345.
[27] K. Lam, E. Chan, H. Leung, M. Au, Concurrency control strategies for ordered data broadcast in mobile computing systems, Information Systems 29 (3) (2004) 207–234.
[28] M. Masud, I. Kiringa, Transaction processing in a peer to peer database network, Data & Knowledge Engineering 70 (4) (2011) 307–334.
[29] K. Liu, V. Lee, A conflict avoidance data allocation algorithm in a multi-channel broadcast environment, Journal of Networks 5 (3) (2010) 343–350.
[30] C. Chen, C. Lee, S. Wang, On optimal scheduling for time-constrained services in multi-channel data dissemination systems, Information Systems 34 (1) (2009) 164–177.
[31] S. Lee, D. Carney, S. Zdonik, Index hint for on-demand broadcasting, in: Proceedings of the 19th International Conference on Data Engineering (ICDE'03), Proceedings of the 19th International Conference on Data Engineering (ICDE'03), vol. 1063, 2003, pp. 726–728.
[32] J. Huang, AIDOA: an adaptive and energy-conserving indexing method for on-demand data broadcasting systems, IEEE Transactions on Systems, Man and Cybernetics, Part A 38 (2) (2008) 331–345.
[33] A. Waluyo, D. Taniar, B. Srinivasan, W. Rahayu, An enhanced global index for location-based mobile broadcast services, Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, IEEE Computer Society, 2010, pp. 1173–1180.
[34] H. Hung, J. Huang, J. Huang, M. Chen, Scheduling dependent items in data broadcasting environments, Proceedings of the 2006 ACM Symposium on Applied Computing (SAC'06), ACM, 2006, pp. 1177–1181.
[35] L. Gao, X. Wang, A game approach for multi-channel allocation in multi-hop wireless networks, Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing, ACM, 2008, pp. 303–312.
[36] J. Lv, V. Lee, M. Li, E. Chen, Profit-based on-demand broadcast scheduling of real-time multi-item requests, Proceedings of the 2010 ACM Symposium on Applied Computing (SAC'10), ACM, 2010, pp. 580–584.
[37] J. Zimmerman, Accounting for decision making and control, Irwin, 1995.
[38] H. Schwetman, CSIM19: a powerful tool for building system models, Proceedings of the 33rd Conference on Winter Simulation, IEEE Computer Society, Washington, DC, USA, 2001, pp. 250–255.
[39] G. Zipf, Human behavior and the principle of least effort: an introduction to human ecology, Addison-Wesley Press, 1949.