Information Sciences 242 (2013) 76–91

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Broadcasting on-demand data with time constraints using multiple channels in wireless broadcast environments

Chuan-Ming Liu *, Ta-Chih Su

Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan

ARTICLE INFO

Article history:
Received 29 August 2010
Received in revised form 3 April 2013
Accepted 19 April 2013
Available online 25 April 2013

Keywords:
On-demand data broadcasting
Data overlapping
NP-hard
Competitive analysis
Latency
Miss rate

ABSTRACT

Data Broadcasting is an effective approach to provide information to a large group of clients in ubiquitous environments. How to generate the data broadcast schedule to make the clients' average waiting time as short as possible is an important issue. In particular, when the data access pattern is dynamic and data have time constraints, such as traffic and stock information, scheduling the broadcast for such data to fulfill the requests is challenging. Since the content of the broadcast is dynamic and the request deadlines should be met, such data broadcasting is referred to as on-demand data broadcasting with time constraints. Many papers have discussed this type of data broadcasting with a single broadcast channel. In this paper, we investigate how to schedule the on-demand broadcast for the data with time constraints using multiple broadcast channels and provide two heuristics to schedule the data broadcast. The objective of the proposed heuristics is to minimize the miss rate (i.e., ratio of the number of requests missing deadlines to the number of all requests) and latency (i.e., time between issuing and termination of the request). We show that the offline version of the considered problem is NP-hard and present a competitive analysis on the proposed heuristics. More discussion about the proposed heuristics is given through extensive simulation experiments. The experimental results validate that the proposed heuristics achieve the objectives.

1. Introduction

Advanced technologies in wireless communications, information systems, and hand-held devices make it possible for mobile clients to conveniently access different kinds of information services, such as electronic news, traffic information, and stock prices. In such an environment, the bandwidth between a server and a client is asymmetric [1,8], that is, the downlink bandwidth is much greater than the uplink bandwidth. When the group of mobile clients is large, the bandwidth of the uplink becomes a bottleneck in traditional client–server model. In wireless environments the number of mobile clients is growing constantly and the information service system is expected to serve an increasingly large number of users. Thus, how to solve the bottleneck problem becomes a critical issue.

In recent years, data broadcasting has been considered an attractive solution for the bottleneck problem and it also provides an efficient way to disseminate the information to a large pool of clients. In general, data broadcasting can be classified into two types [16,21]: *push-based data broadcast* and *on-demand data broadcast*.

* Corresponding author.
E-mail addresses: cmliu@csie.ntut.edu.tw (C.-M. Liu), t7598023@ntut.edu.tw (T.-C. Su).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.04.026

- *push-based data broadcast:* In push-based data broadcast, a server periodically broadcasts data items on a broadcast channel. Fig. 1a illustrates the push-based data broadcast, where the server repeatedly broadcasts data items 1, 2, 3, and, 4 and the clients tune into the channel to access the data items. Most of the scheduling algorithms in this type consider static data access patterns and sequences.
- *on-demand data broadcast:* In on-demand data broadcast, the clients send the requests via an uplink channel and then the server broadcasts the requested data items. Fig. 1b shows the on-demand data broadcast, where the dashed and solid lines represent the uplink and broadcast channels respectively. The on-demand data broadcast can be used for dynamic and large-scale data dissemination.

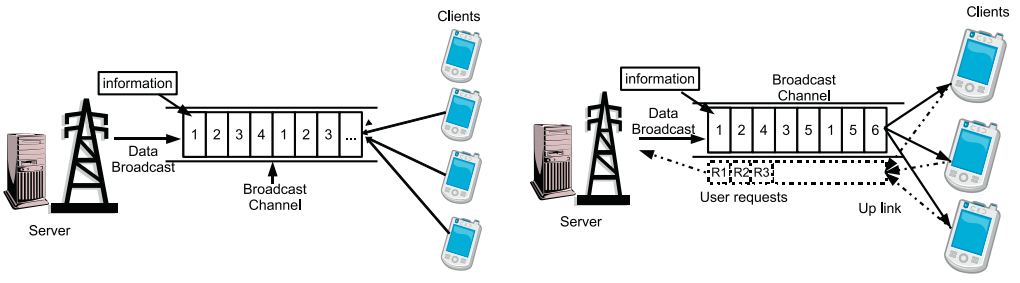
In some information services, data may have temporality, such as traffic information and stock quotes, so the data may become invalid as time passes. To ensure that the data is timely, the clients usually request the data with deadlines and the server then broadcasts the on-demand data before the deadlines. The requested data become invalid when the deadlines are passed. Thus, how to schedule on-demand data broadcasts for data with time constraints is an important topic. To our knowledge, although many papers have discussed this topic, most of them considered a single broadcast channel [3–5,7,12,13,19–21]. For a server, using multiple channels [2,6,15–18,23,24] to provide information makes the broadcast cycle shorter than using one channel, thus serving more users in a short time.

In this paper, we discuss the problem of scheduling on-demand broadcasts for data with time constraints in multi-channel environments. A broadcast consist of a sequence of broadcast slots (or packets). For simplicity, we assume that each data item corresponds to one slot in the broadcast. We consider that each request has its own deadline and has multiple data items associated with it. Data items are related because of the associated requests. If we lose any one of the data items associated with a specific request, the data items received may become useless. For instance, a client may request a Web page which contains some components, such as audio clips or images. When the client uses a browser to access the Web page, the browser will request all the components contained in the Web page after receiving the Web page. Another application would be when a mobile client has an inquiry about the stock prices of some companies. It is highly likely that the mobile client would like to compare the stock information and make a decision on investment, so the quotes of these stocks therefore are related because of the request. In addition, some location-based services, like range query and *k*NN query [9], usually have more than one data item associated with the query.

With multiple broadcast channels, the *data overlap problem* [6] will occur when the requested data items for a request appear in different channels at the same time, and only one channel can be tuned into at a given time instance. More details about the data overlap problem will be given in the next section. The data overlap problem will force the client to wait until the following broadcast cycles to receive the relevant data items, thus leading to a longer latency. However, a data-overlap free broadcast schedule also makes the broadcast cycle long, with many empty slots in the broadcast. In order to reduce the latency, data replication can be used to provide more data items earlier to serve more clients by placing data items in the empty slots. The objective of this paper is to provide on-demand data broadcast schedules where more requests meet their deadlines and reduce the waiting time for the clients. In our design, we avoid the data overlap problem as much as possible and use data replication implicitly to make the broadcast compact and the latency short.

The on-demand data broadcast has following property. The server has no knowledge about all the requests in advance and makes decisions only with the information of the requests already received. This property is similar to the conditions for *online* algorithms [3]. On the contrary, an *offline* algorithm is given the whole problem data from the beginning. The performance of an online algorithm can be measured using *competitive analysis*. An online algorithm is *c-competitive* if the optimal solution generated by the offline algorithm is *c* times better than the solution generated by the online algorithm with the same problem instance.

The rest of this paper is organized as follows. After giving preliminaries and related research in Sections 2 and 3, respectively, we describe the system model in Section 4. The formal definition of the problem and a demonstration that the offline version of the problem is NP-complete are given in Section 5. We then propose two heuristics, MPHH and MPLH, in Section 6.



(a) Push-based data broadcast (b) On-demand data broadcast

Fig. 1. Illustrations of (a) the push-based data broadcast and (b) the on-demand data broadcast: dashed lines indicate that the clients send the requests via an uplink channel; and solid lines denote that the clients retrieve the data from a broadcast channel.

The competitive analysis of proposed heuristics is discussed in Section 7. The simulated experimental results are presented and discussed in Sections 8, and 9 concludes this paper.

2. Preliminaries

Two measures are usually considered when using data broadcasting to provide data items: *latency* and *tuning time*. In this section, we first introduce these two measures. Then we give an another measure, *miss rate*, which is introduced additionally to evaluate the service quality for the requests having deadlines. Last, we present what the data overlap problem is.

2.1. Latency and tuning time

In the past decade, many papers on data broadcasting have been presented using latency and tuning time to evaluate their work [8]. The latency is the time elapsed between issuing and termination of a request and it can be used as an indication for the Quality of Service (QoS) provided by the system. The tuning time is the amount of time that a client spends listening to the broadcast and it can represent the power consumption of mobile users. We use Fig. 2 to illustrate. Suppose that a user requests data 1, 3, and 5. In Fig. 2a, because the client has no knowledge about the position of the data in the broadcast channel, the client needs to always listen to the broadcast channel to access the data. The filled arrows indicate the time slots at which the client tunes into the broadcast channel. The latency and tuning time are both 5. In general, when the broadcast consists of only data, these two cost measures are equivalent. To reduce the energy consumption, one can use an index in the broadcast data. The index allows a mobile client to tune into a broadcast only when data of interest and relevance is available. As shown in Fig. 2b, the latency and tuning time is 6 and 4, respectively, where the unfilled arrows denote the time the client tunes out. With the index, the clients can save energy but has insignificant increase in the latency.

2.2. Miss rate

To evaluate the broadcast schedules when applying data broadcasting to provide information services with time constraint, the *miss rate* is considered. Miss rate is the ratio of the number of requests missing the deadlines to the total number of requests, and it can be expressed as

$$\text{Miss rate} = \frac{\text{Number of requests missing deadlines}}{\text{Total number of requests}} \tag{1}$$

A service with a lower miss rate indicates that the service has a higher quality, and one of our objectives in this work is to generate a broadcast schedule having a lower miss rate.

2.3. Data overlap problems

Recall the *data overlap problem* mentioned in Section 1, which occurs if the following conditions are both simultaneous. The first condition is the assumption that only one channel can be tuned into at a time instance on multiple broadcast channels. The second condition is that there are multiple data items associated with a request. The assumption of the first condition is usually considered in papers relating to data broadcasting with multiple broadcast channels. The data overlap problem occurs when the requested data items for a request are broadcast on different channels at the same time. In this case, only one requested data item can be accessed and the client needs to wait until the following broadcast cycles for the other requested data item(s), thus leading to a longer latency. Fig. 3 shows an example, where the client sends a request associated with data A and B. The data overlap happens at time slot 1. The client can access either A or B at this time slot. In this example, we get data B first and then we wait until time slot 4 to access data A.

Since the on-demand data broadcast can be used for dynamic and large-scale data dissemination, the generated broadcast schedule may not have a regular cycle. Thus, it is hard to predict how long it is necessary to wait until the next broadcast cycle for the other requested data, and this will lead to a higher miss rate. In Fig. 3 if we retrieve data A first, then it is

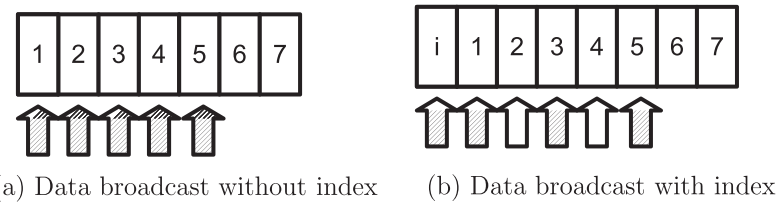


Fig. 2. The latency and tuning time when requesting data 1, 3, and 5, where in (a), the latency and tuning time are 5 respectively and in (b), the latency is 6 but the tuning time is 4; filled arrows indicating the time slots which the client tunes into.

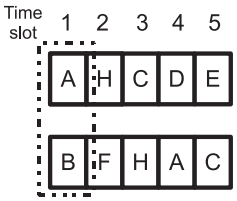


Fig. 3. An example of data overlap problem, where data A and B are associated with a request and overlap at time slot 1.

necessary to wait more than 5 time units to obtain data B, thus having a higher probability to miss the deadline. Therefore, how to prevent the data overlap problem or how to reduce the effect of data overlap problem is an important issue.

3. Related work

In this section, we introduce some related works about data broadcasting. One of the objectives of most papers about data broadcasting is to achieve a short latency [1,8,14,20]. To reduce the latency, the proposed broadcast schedules considered the data access probabilities and/or dependency. Furthermore, some researchers use multiple broadcast channels to have more deduction on the latency [2,6,15,18,23,24]. In general, the algorithms to generate a broadcast schedule having short latency have a tradeoff between complexity and performance. In [23], the push-based data broadcast using multiple broadcast channels was considered, and the authors proposed an approximation algorithm, *GREEDY*, for scheduling data broadcast and showed that its performance is near-optimal. In [24], the authors found that if a popular data item is broadcast more frequently, the generated broadcast schedule can have a shorter latency. An additional constraint, that the data set has a DAG access pattern is considered in [15]. That paper proved that in such situation, the problem is NP-hard and it provided different heuristics considering various data access probabilities.

In on-demand data broadcasting, the author of [7] formulated three measure ingredients, *urgency*, *productivity*, and *fairness*, to ensure the service quality where each request has a time constraint.

1. *Urgency:* Given two data items with the same number of unserved requests, the one with a closer deadline should be broadcast first to reduce the deadline miss rate.
2. *Productivity:* Given two items with the same deadline, the one with more unserved requests should be broadcast first to increase the service satisfaction rate.
3. *Fairness:* When scheduling data items, the broadcast program should be adjustable against dynamic request patterns to avoid service starvation.

Most papers about the on-demand data broadcasting have considered the urgency and productivity of data to ensure the service quality where each request has a time constraint. In [22], the authors applied the earliest deadline first (EDF) policy to schedule the broadcasts, considering the simple condition that a request is associated with only one data item and worked on a single broadcast channel. Thus, some complicated problems, such as data overlap, are not involved. The EDF algorithm gives the data item with earlier deadline a higher priority to be broadcast. For a request associated with multiple data items, all the data items should be received before the deadline of the request, or the request is not fully served and considered a failure. To solve this kind of problem, the *slack time* of a request, is considered. The slack time denotes the maximum number of time slots that the request can be idled or the request will miss its deadline and it represents the urgency of a request when scheduling the broadcast [4,5,7,12,16,19,21]. The Minimal Slack time First (MSF) algorithm is similar to EDF but uses the slack time to decide which data item has a higher priority to be broadcast. In addition to MSF, the Most Requested First (MRF) algorithm is proposed to generate a schedule with a shorter latency. The MRF algorithm assigns a higher priority to the data items that are associated with more requests.

To achieve better performance, many papers have considered both the urgency and productivity of data items to generate a broadcast schedule [4,5,7,12,16,19,21]. In particular, algorithm *SIN-*α (Slack time Inverse Number of pending requests) was proposed to consider both urgency and productivity [21]. Other authors [4,5,12,16] observed that the scheduling algorithms which consider data items independently may cause the request to be unserved when only one or two data items are missing, thus deteriorating the service quality. In other words, the data items received becomes useless for a request which is unserved before its deadline, so the time slots used to broadcast these data items are ineffective and the miss rate increases. Those studies provided solutions to avoid such ineffective broadcast slots by considering data dependency for the requests and thereby increase the productivity of data transfer.

As mentioned, most of the papers related to on-demand data broadcasting have considered only a single broadcast channel. One paper considered multiple channels but did not consider the time constraints of requests [16]. In that paper, the data overlap problem is referred to as the *broadcast mismatch problem* and the proposed Dynamic Urgency Productivity (DUP) algorithm first selects the request having the highest priority. Then from the selected request to broadcast the DUP algorithm chooses a data item that will not induce data overlap problem. In our work, we consider on-demand data

broadcasting using multiple broadcast channels and only one channel can be tuned into at each time instance. Each request has its own deadline and multiple related data, in contrast to [16]. The basic idea of our proposed approach is to derive the result by converting the optimal schedule when enough channels are available. In the next section, we introduce the considered data broadcasting system in detail.

4. System model

The on-demand data broadcasting system we considered is shown in Fig. 4, where a large group of clients can retrieve the data items maintained by a server. On the system, the clients send the requests to the server via an uplink channel. Each request requests some data items and has its deadline. Each requested data item has a unique identifier. The server receives the requests and inserts them into a request queue. When the request queue is not empty, the server schedules the data broadcast according to the requests in the queue and broadcasts the scheduled data items to serve the requests. To generate the broadcast data items at each time instance, the server will check all of the unserved requests. If a request will miss its deadline, the server removes that request from the request queue.

In order to direct the clients to retrieve the relevant data items, we use an additional channel, the *index channel*, to broadcast an index. The rest of the channels are referred to as *data channels*. The index consists of the *id*'s of the data items that will be broadcast on the data channels in the next time slot. For instance, the index i_1 in Fig. 4 contains the *id*'s of data items broadcast in time slot 2. After a client sends the request, it immediately tunes into the index channel to retrieve the index. When a client finds the relevant data items from the index, it tunes into one of the relevant data channels to get the data item. Each data item in the data channels also contains an index to the data items in the next time slot. If the client cannot find the relevant data in the index, it tunes into the index channel in the next time slot. The clients hence can retrieve the relevant data items continuously. If the client retrieves all the related data items before the deadline of the request, we say that the request is *served*.

5. Problems

Suppose that there are c channels and n data items, $d_1, d_2, \ldots, d_n$, in data set D. Each data item is of the same size and takes 1 time slot to be broadcast. On the client side, we assume that there are m requests. Each request Q_i, $1 \leqslant i \leqslant m$, has a deadline dl_i and is associated with a set of q_i data items, $\{d_{i(1)}, d_{i(2)}, \ldots, d_{i(q_i)}\}$, where $d_{i(j)} \in D$, $1 \leqslant j \leqslant q_i \leqslant n$. We denote the total number of requests for a data item d_l as N_{d_l}, $1 \leqslant l \leqslant n$. Set U_i indicates the set of unserved data items in the request Q_i, $1 \leqslant i \leqslant m$, at the current time t_c. If U_i is not empty, the request Q_i is *unserved*. All the unserved requests at the current time t_c form set *UR*. The slack time of the request Q_i at the current time is defined as $dl_i - t_c - |U_i|$ and denoted by $slack_i$. If $|U_i| > 0$ and $slack_i < 0$, then request Q_i will miss its deadline because there are not enough remaining time slots to broadcast the unserved data items in U_i. Suppose the broadcast starts at time t_0, the miss rate after the data items broadcast at time t_j is

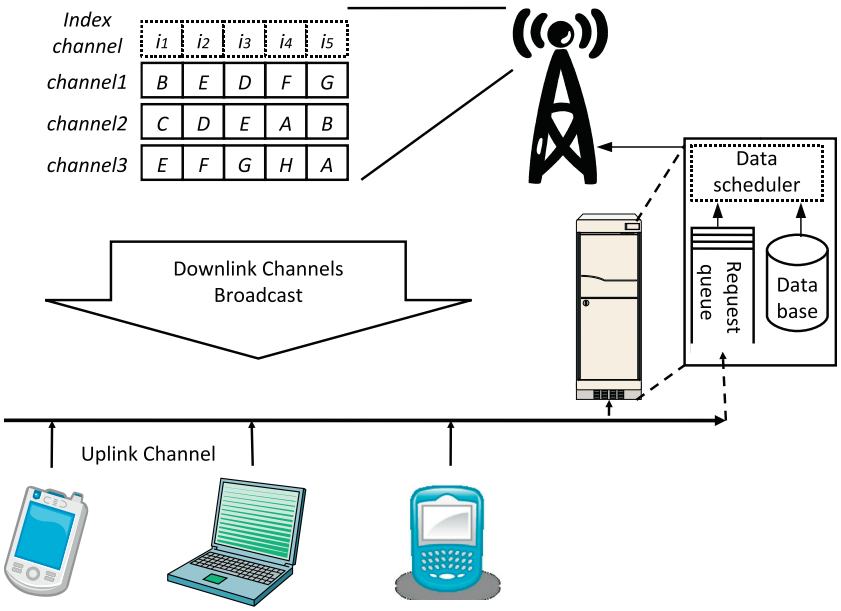


Fig. 4. The architecture of the on-demand data broadcasting system with three broadcast channels: solid squares present the slots in data channels, and dashed squares stand for the indexing channel.

$$R_{(t_j-t_0)} = \frac{|miss(Q_{\{t_j-t_0\}})|}{|Q_{\{t_j-t_0\}}|} \tag{2}$$

where $Q_{\{t_j-t_0\}}$ is the set of the requests received by the server from t_0 to t_j and $|miss(Q_{\{t_j-t_0\}})|$ is the total number of requests which have missed their deadlines at time slot t_j in the set $Q_{\{t_j-t_0\}}$.

In this paper, we first discuss the problem of how to generate a broadcast schedule on multiple channels at each time instance that has the minimum miss rate as determined by Eq. (2). We refer to such a problem as the *On-demand Broadcasting with Minimum Miss rate (OBMM) Problem* and give a formal definition below. In our proposed heuristics, the data replication is implicitly applied for reducing the latency.

Definition 1 (*OBMM problem*). Suppose all the notations are defined as above and the set $RD(t_j)$ consists of the unserved data items of all the requests in $Q_{\{t_j-t_0\}}$ at time t_j. The On-demand Data Broadcasting with Minimum Miss rate Problem is to find a mapping M: $RD(t_j) \to \{1,2,\ldots,c\}$, such that $\sum R_{(t_j-t_0)}$ is minimized.

Since the server has no advance knowledge about the coming requests, the process for scheduling the broadcast is made in an online fashion. We first look at the offline version of the OBMM problem in the following. The competitive analysis for the proposed heuristics will be discussed in Section 7 after the heuristics are introduced. We refer to the off-line version of OBMM problem as *Data Broadcasting with Minimum Miss rate (DBMM) Problem*, and define it as below.

Definition 2 (*DBMM problem*). *Instance:* There are c channels with length L, a set of n data items $D = \{d_1,\ldots,d_n\}$, and a set of m requests $Q = \{Q_1,\ldots,Q_m\}$. Each request Q_i, $1 \leqslant i \leqslant m$, is associated with q_i data items, $d_{i(1)}, d_{i(2)}, \ldots, d_{i(q_i)}$, where $d_{i(j)} \in D, 1 \leqslant j \leqslant q_i \leqslant n$. Furthermore, each request Q_i has its own deadline dl_i and a weight W_i. Every data item needs an unit time to be broadcast.

Question: Does there exist a mapping $f : \bigcup_{i=1}^{m} Q_i \to \{1\cdots c\} \times \{1\cdots L\}$ such that

(1) For two data items $d_{i(s)}$ and $d_{i(t)}$ associated with Q_i, $f(d_{i(s)}) \neq f(d_{i(t)})$,
(2) $\sum_{i=1}^{m} W_i$ is minimized, where W_i is an *indication function* and

$$W_i = \begin{cases} 0, & \text{if for each } d_{i(l)} \text{ associated with } Q_i, f(d_{i(l)}) \leqslant dl_i; \\ 1, & \text{otherwise.} \end{cases}$$

In the definition of DBMM problem, the first objective indicates that the broadcast schedule avoids the data overlapping problem. The second objective is to reduce the miss rate and all of the data items associated some request Q_i should be broadcast before the deadline dl_i. W_i is an indication function used to present if a request is served or not. To show further that the DBMM Problem is NP-complete, we consider a special case of it, where the number of data items associated with each request is the same and equal to the number of channels. The data items associated with different requests are all different. The following gives the definition of the decision problem for the above special case.

Definition 3 (*DBMMρ problem*). *Instance:* There are c channels with length L, an integer K, a set of n data items $D = \{d_1,\ldots,d_n\}$, and a set of m requests $Q = \{Q_1,\ldots,Q_m\}$. Each request Q_i, $1 \leqslant i \leqslant m$, is associated with c data items, $d_{i(1)}$, $d_{i(2)}$, $\ldots$, $d_{i(c)}$, where $d_{i(j)} \in D$, $1 \leqslant j \leqslant c$. Also, each request Q_i has its own deadline dl_i and a weight W_i. In addition, any two data items associated with two different requests are different, and every data item needs an unit time to be broadcast.

Question: Does there exist a mapping $f : \bigcup_{i=1}^{m} Q_i \to \{1\cdots c\} \times \{1\cdots L\}$ such that

(1) For two data items $d_{i(s)}$ and $d_{i(t)}$ associated with Q_i, $f(d_{i(s)}) \neq f(d_{i(t)})$,
(2) $\sum_{i=1}^{n} W_i \leqslant K$, where W_i is the indication function defined as in DBMM problem.

We use a reduction from the *Minimizing the Number of Late Jobs in Unit time Open Shop (MNUOS)* which is NP hard [11] to the DBMMρ Problem to show that the DBMMρ Problem is NP-hard. Thus, the DBMM Problem is NP-hard. The MNUOS problem is defined as follows.

Definition 4 (*MNUOS problem*). *Instance:* There are an integer T, a set of m jobs $J = \{J_1, J_2, \cdots, J_m\}$, and n machines $M = \{M_1, M_2, \ldots, M_n\}$. Each Job J_i consists of n unit operations $o_{i(j)}$, where the jth operation, $1 \leqslant j \leqslant n$, has to be processed on the machine j. Job J_i must be processed in a window defined by a release date $r_i = 1$ and a due date d_i. U_i is a unit penalty of J_i when J_i is failed to be processed before the due date.

Question: Does there exist a schedule f such that

(1) For operations $o_{i(s)}$ and $o_{i(t)}$ in job J_i, $f(o_{i(s)}) \neq f(o_{i(t)})$,
(2) $\sum_{i=1}^{n} U_i \leqslant T$, where $U_i = \begin{cases} 0, & \text{if } \forall o_{i(l)} \in J_i,\ f(o_{i(l)}) \leqslant d_i; \\ 1, & \text{otherwise.} \end{cases}$

Theorem 1. *The DBMMρ problem is NP-complete.*

Proof. It is easy to see that the DBMMρ problem is in NP, since checking an answer simply needs polynomial time. In order to prove the DBMMρ problem is NP-hard, a reduction from the MNUOS problem can be made. Suppose that I' is an instance of the MNUOS problem. A corresponding instance I of the DBMMρ problem can be constructed from I' as follows.

1. An unit operation time is equal to the unit time to broadcast a data item.
2. Let a job J_i correspond to a request Q_i, $1 \leqslant i \leqslant m$ and operations $o_{i(j)}$ in J_i be the data item $d_{i(j)}$ associated with Q_i.
3. Let the n machines be the c channels (*i.e.*, $n = c$).
4. Let U_i be W_i and integer T be the integer K in DBMMρ problem.

Such a construction can be done in polynomial time. It is straightforward to show that there is a solution for an instance I' of the MNUOS problem if and only if there is a solution for the instance I of the DBMMρ problem since the reduction is a one-to-one mapping for the variables from the MNUOS problem to the DBMMρ problem. Hence, the DBMMρ problem is NP-complete. □

Thus, we can conclude the following theorem.

Theorem 2. *The DBMM problem is NP-complete.*

6. Proposed heuristics

As mentioned in Section 3, almost all the related papers considered urgency and productivity in order to have better performance in terms of miss rate and latency. The fairness is generally not considered because avoiding the starvation problem may increase the miss rate and latency of the generated broadcast schedule. We follow these and present two heuristics to generate the broadcast schedules based on the urgency and productivity of data items when the number of channels is limited. The basic idea of the proposed heuristics is to put the broadcast data into c channels as compactly as possible. Consider that there are c channels and the number of unserved requests is m where $c < m$. In our approaches, we first assume that there are m channels and the optimal broadcast schedule can be obtained easily. Then, we convert the resulting m-channel broadcast into another broadcast with c channels and keep a low miss rate and latency. The first heuristic, the *Most Popular First Heuristic (MPFH)*, aggregates popular data items associated with requests efficiently and then selects c data items to be broadcast by considering the urgency and productivity of data items. Such a broadcast schedule, however, may not be optimal in terms of miss rate due to the data redundancy in the generated broadcast. The second heuristic, the *Most Popular Last Heuristic (MPLH)*, is used to remedy this flaw. The idea is to postpone the time to broadcast urgent data in order to serve more requests in a time slot.

Both of the proposed algorithms have three phases: Aggregation Phase, Conversion Phase, and Removing Phase. In the Aggregation Phase, a set of candidate data items is selected to be aggregated according to the popularity of data items. The Conversion Phase will only be executed when the number of candidates is larger than the number of channels. After the Conversion Phase, only c data items are selected to be broadcast in the time slot, and the Removing Phase removes the broadcast data items from the corresponding sets of unserved data items. Note that some data items are overlapped in the broadcast generated by the proposed algorithms. In our design, some data items are repeatedly broadcast in order to reduce the impact caused by avoiding data overlap, thus reducing the latency.

6.1. Most popular first heuristic

In MPFH, the popular data items are selected as the candidates to be broadcast, as follows.

6.1.1. Aggregation phase

For each request Q_j, $1 \leqslant j \leqslant m$, we first select the data item d_i which is associated with the most requests among all the data items associated with Q_j. In other words, d_i is the most popular data item in the data items associated with Q_j. We refer to d_i as the candidate for serving request Q_j and denote it as ca_j. We give two attributes, the slack time and the number of unserved data items of request Q_j, to ca_j. During the aggregation, if two candidates are the same data item, we merge these two candidates into one and the new candidate will retain the smaller slack time and the smaller number of unserved data items. All the candidates are then united into a set CA. For example, suppose there are four candidates, $ca_1 = d_1$, $ca_2 = d_2$, $ca_3 = d_2$, $ca_4 = d_3$ for four requests. After aggregation, the set CA is $\{d_1, d_2, d_3\}$ and the data items in the CA are unique. If the number of data items in CA is smaller than or equal to c, we just put the data items in the broadcast channels directly and proceed to the Removing Phase. Otherwise, the process moves on to the Conversion Phase.

6.1.2. Conversion Phase

In this phase, we select c candidates from CA to broadcast. The selection depends on the urgency and productivity of data items and is determined by the slack time, the number of unserved data items, and the number of associated requests of a

candidate in order. The candidate in *CA* having the smallest slack time will be selected first. If there is a tie, the one with a smaller number of unserved data items will be selected. If exactly one data item still cannot be selected, we select the one having most associated requests. After a data item is selected, it is removed from *CA*. The process is repeated until c candidates are selected. Thus, we can have exactly c data items to broadcast. Please refer to Fig. 5 for algorithmic statements of the rules. Note that if exactly one data item cannot be selected in the end, then one of them can arbitrarily selected, and we select the data item with smaller ID.

6.1.3. Removing Phase

After c candidates are selected, we remove them from the associated unserved data item set. In other words, if candidate ca_j is selected to be broadcast, ca_j is removed from the associated set of unserved data items, U_j. Note that only one data item can be removed from each U_j in the process. If candidates ca_j and ca'_j are selected to be broadcast in time slot t_c and U_j contains both ca_j and ca'_j, we remove only ca_j from U_j. Recall the data overlap problem, where ca_j and ca'_j are overlapping for Q_j at time t_c. Our process leaves ca'_j in the U_i, and ca'_j will be selected as a candidate for Q_j in the following time slots. The data overlap problem is thus resolved. The MPFH continues selecting data items to be broadcast at each time slot until the set of unserved requests, *UR*, is empty. Fig. 5 shows algorithm MPFH.

We now use the example in Fig. 6 to illustrate how MPFH works. There are two channels and three requests, Q_1, Q_2, and Q_3, in the beginning. As shown in Fig. 6a, Q_1 requests three data items, d_1, d_2, and d_3, with the deadline $dl_1 = 6$; Q_2 requests four data items, d_2, d_3, d_4, and d_5, with the deadline $dl_2 = 9$; and Q_3 requests two data items, d_3 and d_4, with the deadline $dl_3 = 3$. The numbers of requests for the data items, $N_{d_1}, N_{d_2}, N_{d_3}, N_{d_4}$, and N_{d_5}, are 1, 2, 3, 2, and 1, respectively.

We first decide the data items to be broadcast at time slot $t = 1$. Data item d_3 will be selected as the candidate for requests Q_1, Q_2, and Q_3 respectively. Then we aggregate the candidates into set *CA*. In this case, there is only one data item d_3 in *CA* after aggregation. The slack time of data item d_3 is $dl_3 - 1 - |U_3| = 0$. Because $|CA| \leqslant 2$, we can directly broadcast d_3 at time slot $t = 1$. In the meanwhile, the process also receives a new request Q_4 consisting of 2 data items, d_3 and d_6, with deadline $dl_4 = 6$. As shown in Fig. 6b, the set of unserved data items for each request at time slot $t = 2$ is $U_1 = \{d_1,d_2\}$, $U_2 = \{d_2,d_4,d_5\}$, $U_3 = \{d_4\}$, and $U_4 = \{d_3,d_6\}$ respectively. N_{d_1}, N_{d_2}, N_{d_3}, N_{d_4}, N_{d_5}, and N_{d_6} now become 1, 2, 1, 2, 1, and 1 respectively. According to our approach, d_2 is selected as the candidate of Q_1 because N_{d_2} is larger than N_{d_1}. In Q_2, because $N_{d_5} < N_{d_2} = N_{d_4}$, we can select any one of d_2 and d_4 as the candidate. In this case, we select the one has the smallest *id*. So, d_2 is selected as the candidate of Q_2. Similarly, we select d_4 and d_3 in Q_3 and Q_4, respectively. We then aggregate the candidates, $ca_1(=d_2)$, $ca_2(=d_2)$, $ca_3(=d_4)$, and $ca_4(=d_3)$ into *CA*. In particular, ca_1 and ca_2 are the same data item. We make a new candidate $ca_{\{1,2\}}$ with slack time $slack_{\{1,2\}} = slack_1$ and the number of unserved data items being $|U_1|$. Hence, after aggregation, *CA* becomes $\{ca_{\{1,2\}}, ca_3, ca_4\}$.

Since $|CA| = 3 > c = 2$, the process moves to the Conversion Phase. The candidate with the minimum slack time, $ca_3(=d_4)$, is first selected to be broadcast and then deleted from *CA*. Then the process continues to select the second data item to be broadcast. *CA* now becomes $\{ca_{\{1,2\}}, ca_4\}$ and $slack_{\{1,2\}}$ and $slack_4$ are both 2. We then select the one having the least $|U_i|$. Because $|U_1| = |U_4| = 2$, we can select either $ca_{\{1,2\}}$ or ca_4. Again, we select the one with the smaller *id*, so $ca_{\{1,2\}}$ is selected, and at time $t = 2$, $ca_3(=d_4)$ and $ca_{\{1,2\}}(=d_2)$ will be broadcast. Fig. 6c shows the result after d_2 and d_4 are broadcast. The unserved data items in each request are $U_1 = \{d_1\}$, $U_2 = \{d_4,d_5\}$, $U_3 = \emptyset$, and $U_4 = \{d_3,d_6\}$, respectively. Because U_3 is empty, we remove Q_3

```
Input: c channels and m unserved requests
Output: the set of at most c data items, SC, to be broadcast
/* Aggregation Phase */
(1) Find the most popular data item as the candidate for each request;
(2) Aggregate the candidates into the set CA;
(3) Set SC = ∅;
(4) if |CA| > c then
    /* Conversion Phase */
    (4.1) for n = 1 to c do
    (4.1.1) Select the candidate ca with the minimum slack time in CA;
            if there is a tie then
              Select the one with the fewest unserved data items;
              if there is a tie then
                Select the one with the most requests associated;
    (4.1.2) SC = SC ∪ {ca};
    (4.1.3) Remove ca from CA;
    else
            SC = CA;
/* Removing Phase */
(5) for each unserved request Q_i, 1 ≤ i ≤ m do
        for each data item d_j ∈ U_i do
            if d_j ∈ SC then
              Remove d_j from U_i;
              Break; /* only one data item can be removed. */
(6) return SC;
```

Fig. 5. A high-level description of the most popular first heuristic.

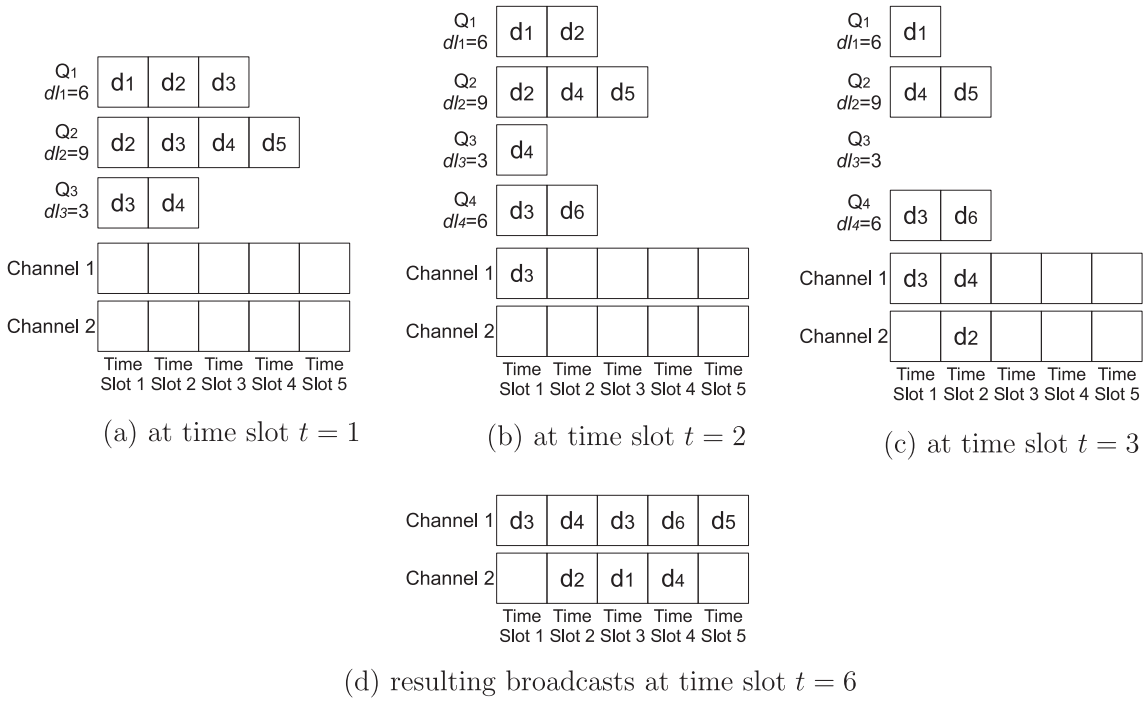


Fig. 6. An example showing the execution of MPFH at time slots 1, 2, and 3 and the resulting broadcasts.

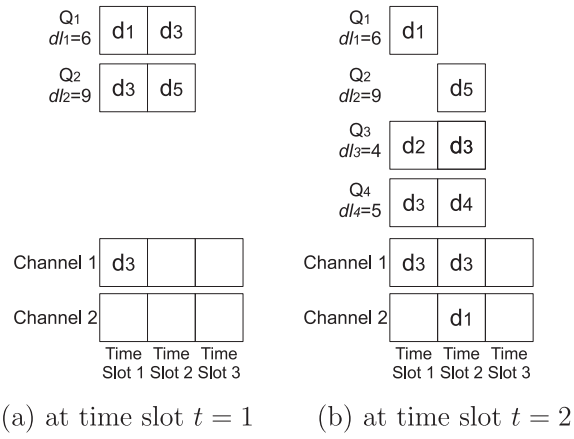


Fig. 7. A broadcast schedule generated by MPFH that can be improved by accumulating more requests for a data item to be broadcast.

from the set of requests. The same process continues until *UR* is empty. Fig. 6d shows the result if there are no other requests received.

6.2. Most popular last heuristic

Algorithm MPFH considers the request urgency and service productivity. Consider the broadcast schedule in Fig. 7 generated by MPFH. Data item d_3 is broadcast at time slot $t = 1$ and $t = 2$ respectively. To serve Q_1 and Q_2 before their deadlines, it is sufficient to broadcast data item d_3 at time slot $t = 2$ once. In this case, d_3 is redundant in the broadcast at time slot $t = 1$. By this observation, we propose the Most Popular Last Heuristic, MPLH, which postpones the time to broadcast popular data in order to serve more requests in the same time slot.

In order to postpone the time to broadcast popular data, the popular data items will not be selected as candidates in Aggregation Phase in algorithm MPLH. A popular data item has a higher probability of being requested by the other clients

in future, so we accumulate more requests for that data item by postponing the time to broadcast it. Hence, the non-popular data items will instead be selected as candidates in algorithm MPLH. The rest of algorithm MPLH is the same as MPFH. The high-level description of algorithm MPLH can be derived by replacing Step (1) in Fig. 5 with

(1) Find the least popular data item as the candidate for each request.

We again use Fig. 6 to illustrate how MPLH works. Data items d_1, d_5, and d_4 will be selected as the candidates for Q_1, Q_2, and Q_3, respectively in Step (1) because d_1, d_5, and d_4 are the data items having fewest associated requests among the data items associated with Q_1, Q_2, and Q_3, respectively. Then we move to Step (2) and aggregate these three candidates into set CA, and CA now has three data items. Because $|CA| = 3 > c = 2$, the process moves to the Conversion Phase. With less slack time, candidates $ca_3(=d_4)$ and $ca_1(=d_1)$ are selected to be broadcast. In the unserved data items for each request in Fig. 8a, d_4 is removed from Q_2 because $ca_2(=d_5)$ is not selected to be broadcast and no data items in Q_2 are removed before. At time slot $t = 2$, candidates $ca_3(=d_3)$ and $ca_1(=d_2)$ will be broadcast. Fig. 8b shows the result after d_3 and d_2 are broadcast and the process at time slot $t = 3$. Note that, candidates $ca_2(=d_5)$ and $ca_4(=d_6)$ are not broadcast at time slot $t = 2$, so we can remove a data item d_i from Q_2 and Q_4 if d_i is broadcast at time slot $t = 2$. We remove d_3 from Q_4. For Q_2, one of d_2 and d_3 can be removed and d_2 is removed in this case. The unserved data items in each request now are $U_1 = \{d_3\}$, $U_2 = \{d_3,d_5\}$, $U_3 = \{\emptyset\}$, and $U_4 = \{d_6\}$, respectively. Because U_3 is empty, we remove Q_3 from request queue. The same process continues until UR is empty. Fig. 8c shows the result if no more requests are received.

7. Competitive analysis

In this section, we present a competitive analysis for our proposed heuristics, MPFH and MPLH. We use OPT to denote the optimal solution for the offline version of OBMM problem in the analysis. For simplicity the number of served requests is considered as the measurement instead of the miss rate. In fact, the miss rate and the number of served requests have the same meaning for the measurement. To derive the competitive ratio, we compare our algorithms with OPT in terms of the number of served requests. Suppose all the notations are as defined in Section 5. In our analysis, we assume that each request is associated with the same number of data items. We then claim the following theorem.

Theorem 3. *Let c be the number of channels and n the total number of data items. Algorithms MPFH and MPLF are $(2-\epsilon)$-competitive in terms of number of served requests, where ϵ is $\frac{c}{n}$.*

Proof. To perform the competitive analysis, we refer to [10] and consider a worst case as the adversary instance, where all requests have the same deadline $\frac{n}{c}$ and each request is associated with k data items, to derive a lower bound on the number of served requests for the proposed algorithms. Note that since the deadline should be larger than k and the number of requests should be larger than c, $n \geqslant k \cdot c$. Without loss of generality, we consider only the generated broadcast schedule in period T with $\frac{n}{c}$ time slots.

The adversary performs as follows. In the beginning (*i.e.*, at time slot 1), the adversary issues $\frac{n}{k}$ requests simultaneously. Each request Q_i requires k different data items, $1 \leqslant i \leqslant \frac{n}{k}$. According to the request identifiers, for simplicity, we can label the data items associated with request Q_i as $d_{(i-1)k+1}, d_{(i-1)k+2}, \ldots, d_{(i-1)k+(k-1)}, d_{ik}$. Then at most c of them can be being served at time slot 1, say $Q_1, \ldots, Q_c$. Without loss of generality, we can assume that data items $d_1, d_{k+1}, \ldots, d_{(c-1)k+1}$ are broadcast at time slot 1 on c channels for requests $Q_1, Q_2, \ldots, Q_c$, respectively. Afterwards, at each time slot, the adversary issues another c

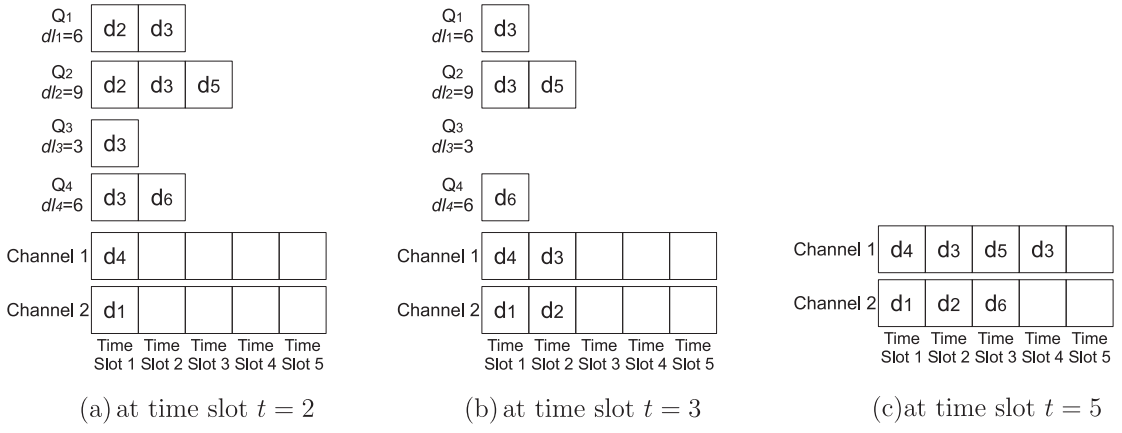


Fig. 8. An example of the execution of MPFH at time slots 2 and 3 and the resulting broadcast schedule for the case in Fig. 6.

requests, which are the same as the requests being served at time slot 1. The same but new c requests are issued repeatedly. Because the associated data items of the new coming requests have been broadcast at the previous time slot, it will be difficult for the new requests to be served before their deadlines. We now discuss how the proposed two algorithms work for the adversary instance.

MPFH

Algorithm MPFH schedules the data items according to the urgency and productivity of data items. The worst case in terms of the number of served requests occurs when the urgency and productivity of data items are the same. For the adversary instance, there are $\frac{n}{k}$ requests at time slot 1. After Aggregation Phase and Conversion Phase, c data items are selected. Suppose that the data items selected are $d_1, d_{k+1}, \cdots, d_{(c-1)k+1}$ as shown in Fig. 9a and the corresponding requests to these selected data items are thus $Q_1, Q_2, \ldots, Q_c$ according to our labeling. Each corresponding request then has $k-1$ unserved data items and each remaining request Q_i, $c+1 \leqslant i \leqslant \frac{n}{k}$, has k unserved data items respectively.

At time slot 2, the adversary issues another c requests that are the same as the requests being served at time slot 1. There are now totally $\frac{n}{k}+c$ unserved requests and the unserved data items associated with the requests being served at time slot 1 have a higher priority to be broadcast according the algorithm MPFH because of their popularity. After the scheduling process, the data items selected to be broadcast are $d_2, d_{k+2}, \ldots, d_{(c-1)k+2}$. The same process continues until time slot k. At time slot k, the requests being served at time slot 1 will be completely served as Fig. 9b shows. Furthermore, according to the heuristic used in MPFH, all the requests issued from time slot 2 to time slot $\frac{n}{c}-(k-1)$ can be served completely. In other words, the requests after time slot $\frac{n}{c}-(k-1)$ cannot be served. So, by including the c served request at time slot 1, the total number of served requests by MPFH is $c \times \left(\left(\frac{n}{c}\right)-(k-1)\right)$.

On the other hand, for the offline case where all the requests issued at each time slot are known in advance, the optimal solution, *OPT*, can be achieved by serving the requests remaining in time slot 1 (*i.e.*, $Q_{c+1}, \cdots, Q_{\frac{n}{k}}$) at proper time slots since the requests after time slot $\frac{n}{c}-(k-1)$ cannot be fully served. The best number of served requests for the optimal solution thus can be $\frac{n}{k}+c\left[\frac{n}{c}-(k-1)-1\right]=\frac{n}{k}+n-c\times k$. So the competitive ratio of MPFH to OPT on the lower bound is $\frac{\frac{n}{k}+n-c\times k}{c\times\left(\left(\frac{n}{c}\right)-(k-1)\right)}=2-\frac{c}{n}$ in terms of the number of served requests.

MPLH

Algorithm MPLH also schedules the data items according to the urgency and productivity of data items, but the generated broadcast schedule is different when considering the same adversary instance because MPLH selects the unpopular data items to be broadcast. In the resulting broadcast schedule, algorithm MPLH will repeatedly broadcast the same data items that are broadcast at time slot 1.

Initially, there are $\frac{n}{k}$ requests and all of these requests have the same priority. At time slot 1, after Aggregation Phase and Conversion Phase, c data items, say $d_1, d_{k+1}, \ldots, d_{(c-1)k+1}$, are selected as shown in Fig. 10.

At time slot 2, according to their popularity, the candidates to be served for all the requests include the candidates suggested by the requests received in time slot 1 and $d_1, d_{k+1}, \ldots, d_{(c-1)k+1}$ which are selected by the new coming requests. In the Conversion Phase, the candidates suggested by the requests received in time slot 1 will not be selected since they have a longer slack time. The remaining candidates all have the same priority to be selected. According to the policy of MPLH, the tied candidates with smaller id will be selected. So, again, data items $d_1, d_{k+1}, \ldots, d_{(c-1)k+1}$ are selected to be broadcast.

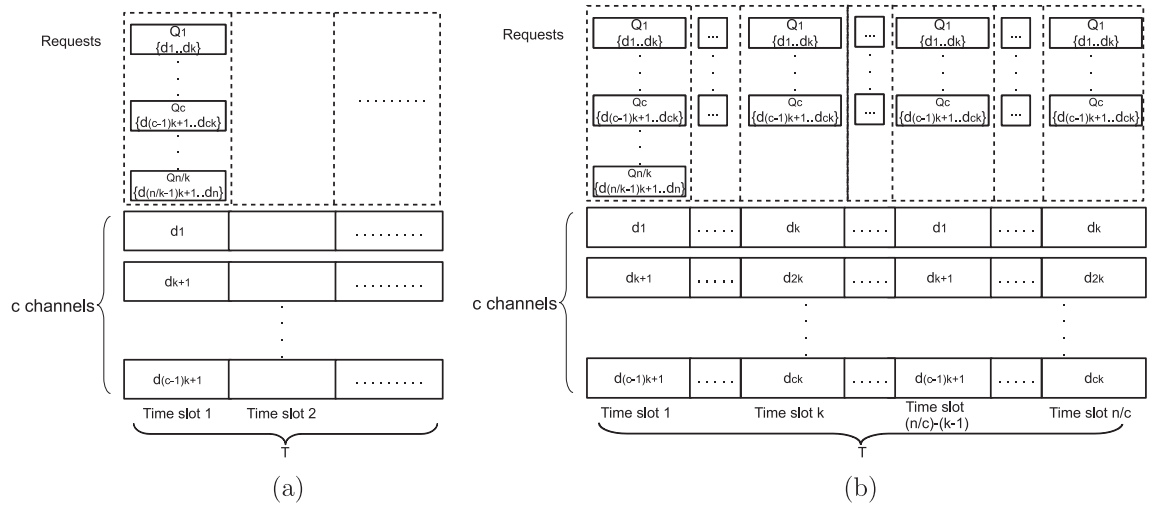


Fig. 9. Illustration of MPFH at (a) time slots 1 and (b) time slot k, where T is the considered period.

The same process proceeds and at each time slot between *2* and $\frac{n}{c}$, the adversary generates c requests which are the same as the ones being served at time slot 1. MPLH repeatedly broadcasts the same data items $(d_1, d_{k+1}, \ldots, d_{(c-1)k+1})$ from time slot *1* to $\frac{n}{c} - (k-1)$. Afterwards, at each time slot the new requests cannot be served completely and the previous $c \times ((\frac{n}{c}) - (k-1))$ requests can be completely served. Please refer to Fig. 10. Hence, the number of served requests by MPLH is $c \times ((\frac{n}{c}) - (k-1))$. Again, in comparison with OPT, the competitive ratio of MPFH to OPT on the lower bound is $\frac{\frac{n}{k}+n-c\times k}{c\times\left(\left(\frac{n}{c}\right)-(k-1)\right)} = 2 - \frac{c}{n}$. □

8. Experiments

This section presents the results of the simulation experiments and our findings. We compare our proposed algorithms with three other algorithms, EDF, MSF, and DUP. The DUP algorithm has good performance for on-demand data broadcasting in multi-channel environments without considering the deadlines of requests. Recall that EDF and MSF are the scheduling algorithms for a single broadcast channel. In order to use these two algorithms on multiple broadcast channels, we modify them with the reduction on the impact caused by data overlap problem. In EDF and MSF, all the data items associated with the selected requests are placed on the channels having the minimum length. Two metrics, miss rate and latency, are used to evaluate the performance of the compared algorithms, including our proposed heuristics, MPFH and MPLH.

All the parameters used in our simulation are shown in Table 1. In the simulation, when the server broadcasts data items in a time slot, it also collects the new requests at the same time. Then the server processes all the requests it has received for the coming time slot. The arrival rate of requests in one time slot is uniformly distributed in the range of [10,15]. The number of data items associated with each request ranges uniformly from 7 to 14. The deadline of a request is given by: $t_c +$ ($\alpha \times$ request length), where t_c is the current time, *request length* is the number of associated data items of a request, and α is selected uniformly from the range of [2,10]. The access pattern of data items is uniform distribution or Zipf distribution [25] with parameter *Zipf* = 0.8. In the Zipf distribution, data item d_1 is the most frequently accessed data item, while the last data item is the least frequently accessed. For DUP algorithm, the request productivity is set to 0.5 since such a setting has a better performance shown in the experiments of [16]. We assume that the time spent to process the data items to be broadcast for each time slot is less than one time slot. For each number of channels, we execute 10,000 requests in a round. The reported result is the average of 100 rounds.

In these experiments, we assume that there are no more requests received by the server after some time slot t_e and the unserved requests in $Q_{\{t_e-t_0\}}$ at time slot t_e will be continuously processed by the server until they are completely served or have missed their deadlines. In order to decrease the effect of these requests on measuring the miss rate, we will extend the time to measure the miss rate. The latency is measured only when the request deadline is satisfied.

8.1. Comparisons on miss rate

Fig. 11 shows the miss rates using algorithms MPFH, MPLH, EDF, MSF, and DUP to process 10,000 requests with different database sizes, respectively, on different numbers of channels. Two different access patterns are included. The results using uniform access patterns are presented by MPFH-uni, MPLH-uni, EDF-uni, MSF-uni, and DUP-uni, respectively and shown

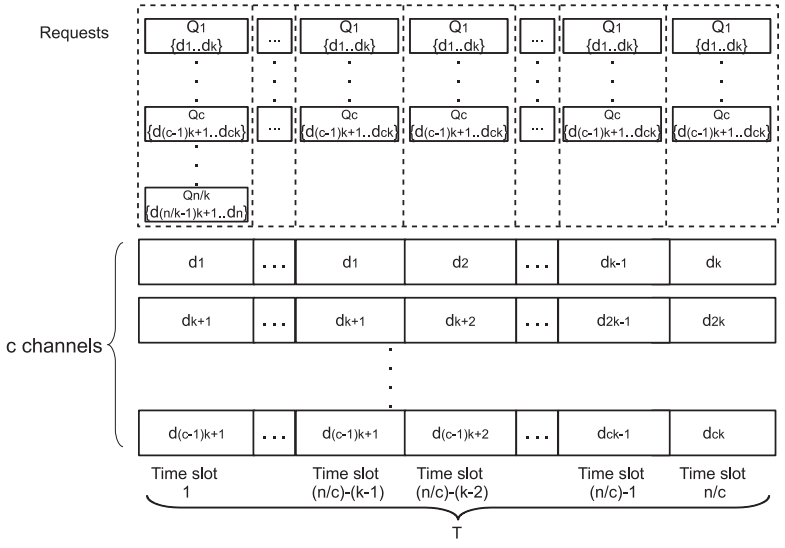


Fig. 10. Illustration of MPLH for the adversary instance at some time slots, where T is the considered period.

Table 1
Parameters and the corresponding values used in the simulation.

Parameter	Value
Database size	100, 200, 500, 1000
Size of a data item	Fitting in one time slot
# of Channels	2–10
Arrival rate (# of requests/time slot)	Uniform distribution [10,15]
# of Associated data items per request	Uniform distribution [7,14]
Deadline of a request	$t_c + (\alpha \times$ request length), α = uniform [2,10]
Data access pattern	Uniform, Zipf = 0.8

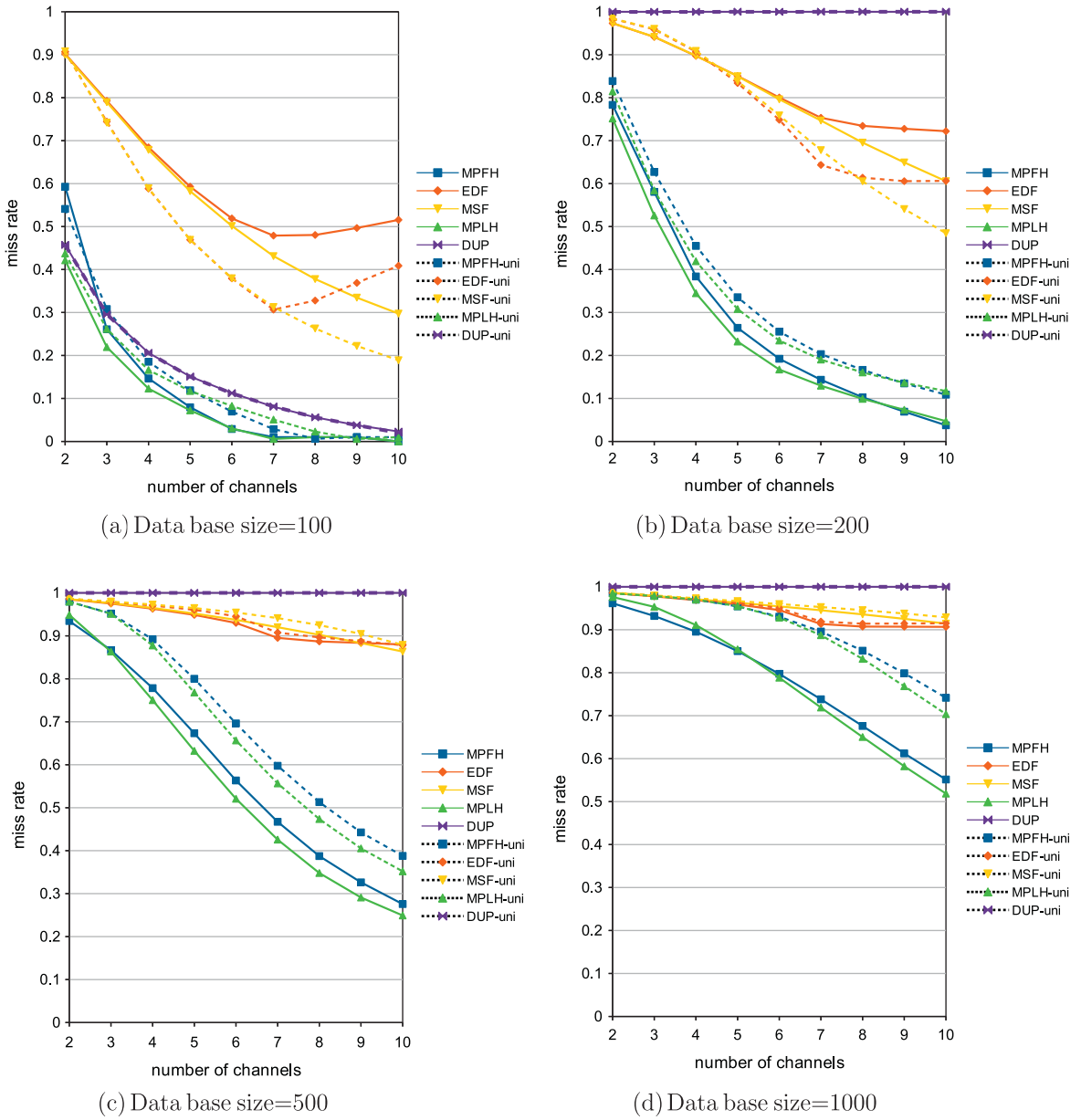


Fig. 11. Comparison of miss rates for MPFH, MPLH, DUP, EDF, and MSF, when the database sizes are (a) 100, (b) 200, (c) 500, and (d) 1000, with different numbers of channels.

with dashed lines. The results show that our algorithms MPFH and MPLH perform better than EDF and MSF in term of miss rate. Especially, when the size of database is small, MPFH and MPLH can almost serve all of the requests before their deadlines as the number of channels increases. DUP performs between our proposed heuristics and the two adapted ones when the database size is small. With the larger database, the miss rate of DUP approaches 100% for both data access distributions. With Zipf distribution, MPFH and MPLH have a lower miss rate than the ones with uniform distribution. The similarity of the associated data items between requests is higher in Zipf distribution and our algorithms aggregate the associated data items for requests when scheduling the data items, so the broadcast schedules can serve more requests. The trend is opposite for EDF and MSF because neither of them consider the aggregation of associated data items. DUP has a similar miss rate for uniform and Zipf distributions since the deadlines of the requests are not considered and the requests can be easily unserved.

We now discuss MPLH and MPFH. With a small number of channels, MPLH performs better than MPFH when the database size is small as shown in Fig. 11a and b. Because the number of channels is few, the impact of broadcasting redundant data items in MPFH on the miss rate is greater. The miss rate of the broadcast schedule generated by MPFH is therefore larger

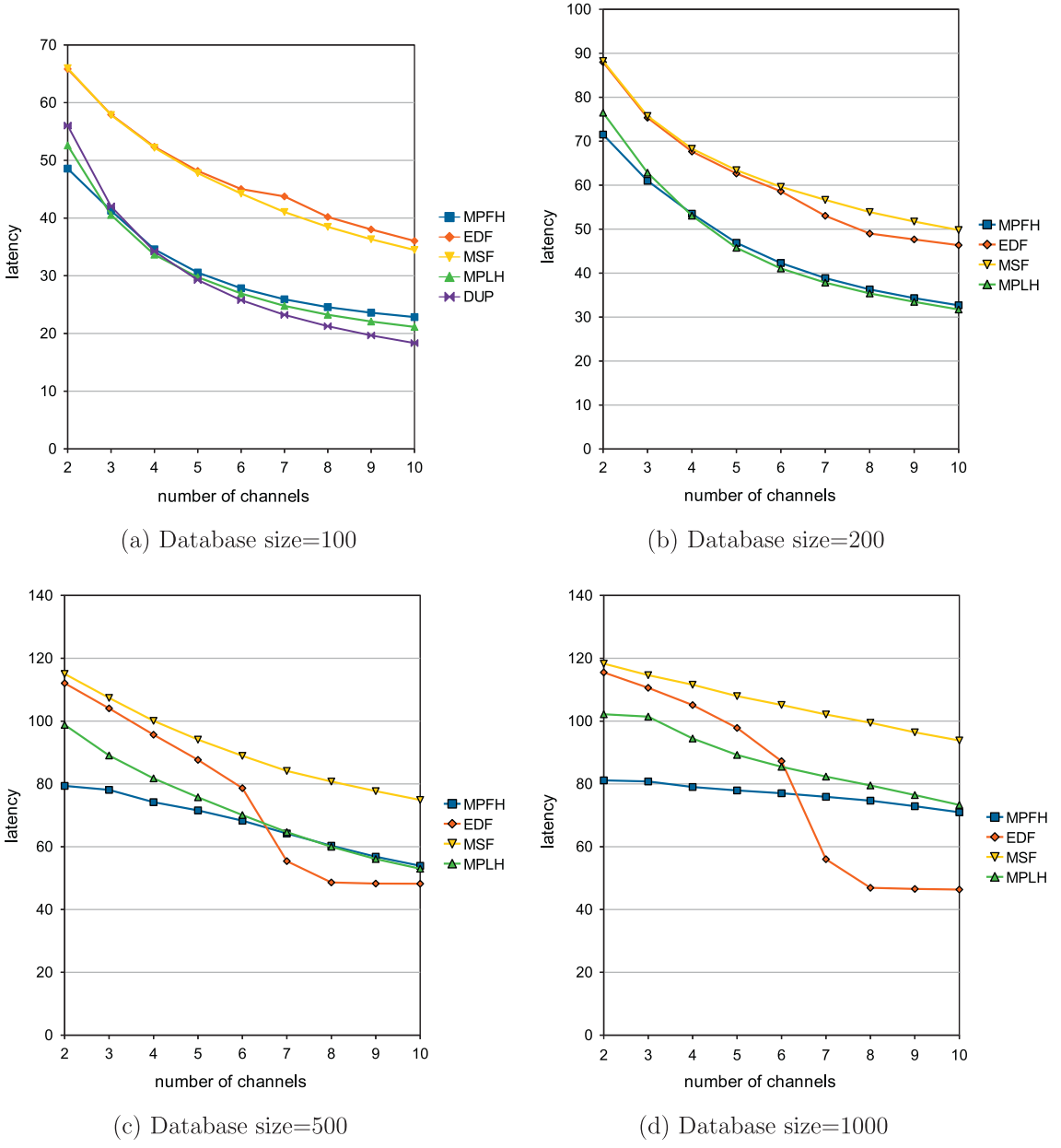


Fig. 12. Comparison of the latency for MPFH, MPLH, DUP, EDF, and MSF, when the database sizes are (a) 100, (b) 200, (c) 500, and (d) 1000, with different numbers of channels, where the latency of DUP is not considered due to high miss rate when the database size equals to (b) 200, (c) 500, and (d) 1000, respectively.

than the one generated by MPLH. Recall that MPLH postpones the time to broadcast hot data items in order to server more requests in the same time slot. Thus, the number of redundant data items broadcast by MPLH decreases and more requests can be served. However, as the number of channels increases, MPFH will do better instead and the server can broadcast more data items with more channels. So, the impact of broadcasting redundant data items in MPFH becomes less. In other words, the server has enough channels to tolerate broadcasting redundant data items.

When the database size increases, as shown in Fig. 11c and d, the trend is opposite and MPFH performs better than MPLH with a small number of channels. We conjecture that the similarity between requests becomes low due to the larger database size and since the number of channels is small, the effectiveness of postponing data items disappears. With more detail, in MPLH, more data items are postponed to be broadcast and the number of associated requests of a postponed data item is small. Thus, some urgent data items will be abandoned when they have to be broadcast. The server broadcasts too many non-urgent data items and the urgent data items are not selected to broadcast when the number of channels is small, thus resulting in a higher miss rate. When the number of channels increases, more data items can be broadcast in a time slot. The postponement in MPLH leads to broadcast fewer redundant data items. Thus, MPLH has better performance. These results match the fundamental ideas of MPFH and MPLH, respectively. MPFH efficiently aggregates data items associated with requests, but it broadcasts redundant data items frequently. Thus, MPFH can perform better when the conditions are beneficial to aggregation and broadcasting redundant data items can be tolerated. Again, the miss rate of DUP is worse than MPFH and MPLH because the deadlines of the requests are not considered in DUP.

8.2. Comparisons on latency

Fig. 12 compares the latency among algorithms MPFH, MPLH, EDF, MSF, and DUP. We discuss only the data access pattern in Zipf distribution because the data access pattern in uniform distribution shows the similar trends. Since the miss rate of DUP is almost 100% when database size is larger than 200, the number of served requests to be used in measuring the latency is insignificant. DUP is thus not included in the plots for the database sizes 200, 500, and 1000. As shown in Fig. 12a where the database size is 100, the three algorithms for multiple channels, MPFH, MPLH, and DUP, have a similar latency, which is shorter than the other two. In comparison, MPFH, MPLH, and DUP have lower miss rates than EDF and MSF. As the database size increases (Fig. 12b–d), MPFH performs better than MPLH. Because the miss rate is high and the similarity between requests is low, more requests can be served in an earlier stage by the broadcast generated by MPFH without postponement. Thus, MPFH has a shorter latency. Notice that, the latency of EDF intensely descends when the number of channels and the database size increase. Recall that the latency is measured only when the deadline of the request is satisfied. Algorithm EDF generates the broadcast using the earliest deadline first policy and it has a high miss rate, so EDF can have a shorter average latency on the served requests.

9. Conclusions

In this paper, we discuss how to generate a broadcast schedule for on-demand data broadcast with time constraint on multiple channels. The problem is formulated and referred to as the On-demand Broadcasting with Minimum Miss rate (OBMM) Problem. This is an online problem and we show that the offline version of the OBMM problem is NP-Hard. Two heuristics are provided for the problem: algorithm MPFH considers the popular data items first and algorithm MPLH tries to serve more requests by broadcasting the popular data items late. Both algorithms also consider the urgency and productivity of data items. In the design, the impact of data overlap problem is reduced and data replication is applied implicitly to reduce the latency.

Since OBMM problem is an online problem, we perform a competitive analysis for the proposed algorithm and show that these two proposed algorithm are $2-\epsilon$-competitive, where $\epsilon=\frac{c}{n}$, c is the number of channels, and n is the number of data items, in terms of the number of served requests. The experimental results show that algorithms MPFH and MPLF perform well in terms of miss rate and latency and follow the principles of algorithms MPFH and MPLH. Each algorithm has better performance than the other in some specific conditions. MPLH generates a broadcast schedule with lower miss rate when the number of channels and the size of database are both either small or large. In general, the broadcast schedule generated by MPFH leads to a shorter latency than MPLH when the database size is large.

In the future, we will consider the tuning time when designing the algorithms for generating the broadcast schedules. We plan to provide an advanced index such that mobile clients can retrieve the data item efficiently and save power. In addition, we will investigate coding techniques for the on-demand data broadcast, since encoding the data items can decrease the size of datasets, yielding a shorter latency and lower miss rate. Furthermore, using a coding technique can help clients to recover missing data items, which usually occur in wireless data broadcasting environments.

References

[1] S. Acharya, R. Alonso, M. Franklin, S. Zdonik. Broadcast disks: data management for asymmetric communication environments, in: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, 1995, pp. 199–210.
[2] S. Anticaglia, F. Barsi, A.A. Bertossi, L. Iamele, M.C. Pinotti, Efficient heuristics for data broadcasting on multiple channels, Wireless Networks 14 (2) (2008) 219–231.
[3] J. Chang, T. Erlebach, R. Gailis, S. Khuller, Broadcast scheduling: algorithms and complexity, ACM Transactions on Algorithms 7 (4) (2011) 1–14.

[4] J. Chen, G. Huang, V.C.-S. Lee, Scheduling algorithm for multi-item requests with time constraints in mobile computing environments, in: Proceedings of the 2007 International Conference on Parallel and Distributed Systems, vol. 2, 2007, pp. 1–7.
[5] J. Chen, V.C.-S. Lee, E. Chan, Scheduling real-time multi-item requests in wireless on-demand broadcast networks, in: Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology, 2007, pp. 125–131.
[6] S.-Y. Fu, C.-M. Liu, Broadcast schedules and query processing for K nearest neighbors search on multi-dimensional index trees in a multi-channel environment, in: Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics, vol. 3, 2006, pp. 2646–2651.
[7] C.-L. Hu, On-demand real-time information dissemination: a general approach with fairness, productivity and urgency, in: Proceedings of the 2007 International Conference on Advanced Information Networking and Applications, 2007, pp. 362–369.
[8] T. Imieliński, S. Viswanathan, B.R. Badrinath, Data on air: organization and access, IEEE Transactions on Knowledge and Data Engineering 9 (3) (1997) 353–372.
[9] H. Jung, Y.D. Chung, L. Liu, Processing generalized k-nearest neighbor queries on a wireless broadcast stream, Information Sciences 188 (2012) 64–79.
[10] J.-H. Kim, K.-Y. Chwa, Scheduling broadcasts with deadlines, Theoretical Computer Science 325 (3) (2004) 479–488.
[11] S.A. Kravchenko, On the complexity of minimizing the number of late jobs in unit time open shop, Discrete Applied Mathematics 100 (1-2) (2000) 127–132.
[12] K.-W. Lam, V.C.-S. Lee, X. Wu, On-demand broadcast for mobile real-time multi-item requests, in: Proceedings of the 2006 International Conference on Computing Informatics, 2006, pp. 1–6.
[13] C. Liaskos, A. Xeros, G.I. Papadimitriou, M. Lestas, A. Pitsillides, Broadcast scheduling with multiple concurrent costs, IEEE Transactions on Broadcasting 58 (2) (2012) 178–186.
[14] C.K. Liaskos, S.G. Petridou, G.I. Papadimitriou, P. Nicopolitidis, A.S. Pomportsis, On the analytical performance optimization of wireless data broadcasting, IEEE Transactions on Vehicular Technology 59 (2) (2010) 884–895.
[15] K.-F. Lin, C.-M. Liu, Broadcasting dependent data with minimized access latency in a multi-channel environment, in: Proceedings of the 2006 international conference on Wireless Communications and Mobile Computing, 2006, pp. 809–814.
[16] K. Liu, V.C.S. Lee, On-demand broadcast for multiple-item requests in a multiple-channel environment, Information Sciences 180 (22) (2010) 4336–4352.
[17] J. Lv, V.C.S. Lee, M. Li, E. Chen, Profit-based scheduling and channel allocation for multi-item requests in real-time on-demand data broadcast systems, Data & Knowledge Engineering 73 (2012) 23–42.
[18] W.-C. Peng, M.-S. Chen, Dynamic generation of data broadcasting programs for a broadcast disk array in a mobile computing environment, in: Proceedings of the 9th International Conference on Information and knowledge Management, 2000, pp. 38–45.
[19] S.K. Udgata. A dynamic, real-time and on-demand heuristic broadcasting scheme for multiple data-item transactions in wireless environment, in: Proceedings of the Fourth International Conference on Wireless Communication and Sensor Networks, 2008, pp. 40–44.
[20] J.-Y. Wang, Set-based broadcast scheduling for minimizing the worst access time of multiple data items in wireless environments, Information Sciences 199 (2012) 93–108.
[21] J. Xu, X. Tang, W.-C. Lee, Time-critical on-demand data broadcast: algorithms, analysis, and performance evaluation, IEEE Transactions on Parallel and Distributed Systems 17 (2006) 3–14.
[22] P. Xuan, S. Sen, O. Gonzalez, O. Gonz Alez, J. Fernandez, K. Ramamritham. Broadcast on demand: efficient and timely dissemination of data in mobile environments, in: Proceedings of the 3rd IEEE Real-Time Technology Application Symposium, 1997, pp. 38–48.
[23] W.G. Yee, S.B. Navathe, E. Omiecinski, C. Jermaine, Efficient data allocation over multiple channels at broadcast servers, IEEE Transactions on Computers 51 (10) (2002) 1231–1236.
[24] S.-Y. Yi, S. Nam, S. Jung, Effective generation of data broadcast schedules with different allocation numbers for multiple wireless channels, IEEE Transactions on Knowledge and Data Engineering 20 (5) (2008) 668–677.
[25] G.K. Zipf, Human Behavior and the Principle of Least Effort, Addison-Wesley, Massachusetts, 1949.