An Efficient Partition and Matching Algorithm for Query-Set-based Broadcasting in Multiple Channel Mobile Environment

Jing-Feng Lin[1], Guang-Ming Wu[2], and Derchian Tsaih[3]
[1] Department of Computer Science and Information Engineering, Nan-Hua University
[2] Department of Information Management, Nan-Hua University
[3] Department of Electronic Commerce Management, Nan-Hua University
32,Chung Keng Li, Dalin, Chiayi, Taiwan
jflin@mail.nhu.edu.tw, gmwu@mail.nhu.edu.tw, dctsaih@mail.nhu.edu.tw,
http://www.nhu.edu.tw

ABSTRACT: -Broadcast is an efficient and scalable method for resolving the bandwidth limitation in a wireless environment. In many applications, mobile clients might need more than one data item. However, most previous researches on query-set-based broadcasting are restricted to a single broadcast channel environment. The results are limited applicability to the upcoming mobile environments. In this paper, we relax this restriction and explore the problem of query-set-based broadcasting in multiple broadcast channels. In multi-channel query-set-based broadcasting, we discover data collision (two data items in the same query set are arranged on two channels at the same time slot) is an important factor to affect users' access time. In this paper, we introduce the new data collision problem motivated by multi-channel query-set-based broadcasting environment. We then present a two-stage scheme of data partitioning and data matching to solve the new data collision problem. Experiments are performed to justify the benefit of our approach.

Key-Words: - Access time, data broadcast, mobile environment, multi-channel, query-set-based broadcasting.

1. Introduction

Recent advances in computer hardware technology have made possible the production of small computers, like PDAs and notebooks, which can be carried around by users. These small computers can be equipped with wireless communication devices that enable users to access global data services from any location. In a wireless environment, there are two kinds of communications between a server and the mobile clients. One is broadcasting and the other is request-and-reply. Due to restrictions on bandwidth and energy, in wireless environment the broadcasting method is preferred [11].

Two important factors must be considered in a broadcast-based information system, access time [1, 3, 13, 16, 21, 24, 25] and tuning time [5, 7, 10, 17]. The access time is the time elapsed from the moment a client device submits a query into the broadcast channel to the moment the desired data are acquired. This is the total time a client device must spend and is often used to evaluate the performance of the broadcast system. The tuning

time is the time spent by the client listening to the broadcast channel. When the clients are listening to the data in the broadcast channel, the clients are in the active mode. Therefore, the tuning time is often used to evaluate the power consumption of the clients. The aim of our paper is to reduce the access time through intelligent organization of the broadcast data.

Many approaches have been proposed to reduce the access time [1, 2, 4, 8, 9, 14, 18, 20]. In these papers, a single broadcast channel is used to broadcast data items in different frequencies according to their relative access rates. Note that, there exist situations where multiple low-bandwidth physical channels cannot be combined into a single high-bandwidth physical channel [19, 22]. In the multi-channel environment, the system can schedule data items on multiple channels [19, 22]. In addition, several network standards, such as FDMA-based systems, divide the network bandwidth into several physical channels where individual mobile clients listen to one channel at a time.

Furthermore, most of the previous approaches assume that each mobile client needs only one data item. They do not consider the relationship between data objects when a query contains more than one data item. However, in many situations, a mobile client might need more than one item of data. [3, 8, 15, 23] proposed scheduling methods for single channel query-set-based data broadcasting. These works studied on broadcasting dependent data are restricted to a single broadcast channel environment. The results are limited applicability to the upcoming mobile environment. To view of this, in this paper, we explore the query-set-based broadcast problem in multiple broadcast channels. In multi-channel mobile environment, the data items must be scheduled in the channels. *Data collision* occurred when two data items are transmitted to a mobile client on two different channels at the same time slot. Since the mobile client can only listen to a channel at the same time to access one of the data items, data collision forces the mobile client to wait for another until next broadcast cycle. This leads to an increase of the client's access time and the length of the access time are certain to exceed one broadcast cycle length. To reduce the number of data collisions, an intelligent idea is partitioning the data items which are in same query set into the same channel. Therefore, this is an important work to develop a good partition method to reduce the *dependent data items* (data items in the same query set) into different channels.

In this article, we introduce a new data collision problem motivated by multi-channel mobile environment. For the data collision problem, we investigate an efficient partitioning method. Our partitioning algorithm can partition all data items into K channels and reduce the data collision probability. Following the partitioning, we propose a matching based method to decide which data items (partitioned in different channels) will be broadcasted in the same time slot and minimize the number of data collisions. In the matching algorithm, we first transfer the matching problem into a maximum flow problem. Next, we applied the maximum bipartite matching technique which derived from the Ford-Fulkerson method [6] to reduce the number of data collisions. Experimental results show that our algorithm can reduce the number of data collisions required for queries efficiently.

The remainder of this paper is organized as follows. Section 2 formulates the data collision problem in broadcast environment. Section 3 proposes our two-stage algorithm. Section 4 reports the experimental results. Finally, conclusions are given in Section 5.

2. Problem Formulation

Figure 1 shows an example system architecture of a data broadcast system which broadcasts data items periodically according to a broadcast scheduling. We assume that there are K channels in a broadcast area, each denoted C_i, $1 \le i \le K$. A database is made up of N unit-sized items, denoted d_j, $1 \le j \le N$. Each item is broadcasted on one of the channels, so C_i broadcasts N_i data items $1 \le i \le K$, $\sum_{i=1}^{K} N_i = N$. Let L be the length of the broadcast program. L is equal to $\left\lceil \frac{N}{K} \right\rceil$. We assume that $N = K \times L$ without loss of generality. Each channel broadcasts its data items periodically according to a broadcast scheduling. Please see figure 1. A broadcast scheduling is an arrangement of all data items D on K broadcast channels and every data item appears in exactly one position on one of the channels.

Let $Q = \{q_1, q_2, \ldots, q_M\}$ be a set of M query patterns whose data set and frequency (or weight) are denoted by $QDS(q_i)$, and $freq\ (q_i)$, $1 \le i \le M$. Let $D = \{d_1, d_2, \ldots, d_N\}$, denote the union of the data items of Q, that is, $D = \bigcup_{1 \le i \le M} QDS(q_i)$.

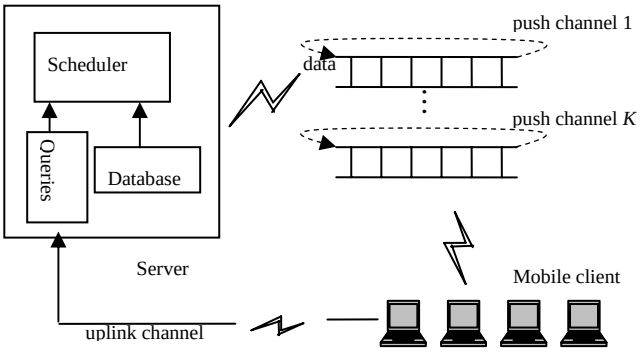

Fig. 1: An architecture of a multi-channel wireless broadcasting system.

Example 1: As shown in figure 2, consider a database D containing eight data items, $D = \{d_1, d_2, \ldots, d_8\}$. Assume that there are two channels on server. Data items d_1, d_2, d_5, and d_8 are partitioned into channel 1 and d_3, d_4, d_6, and d_7 are partitioned into channel 2. The broadcast ordering of channel 1 is $<d_1, d_5, d_8, d_2>$ and channel 2 is $<d_4, d_7, d_6, d_3>$. There are four queries in Q, $Q = \{q_1, q_2, q_3, q_4\}$. Suppose $QDS(q_1)=\{d_1, d_4\}$, $QDS(q_2)=\{d_2, d_6\}$, $QDS(q_3)=\{d_5, d_8\}$, and $QDS(q_4)=\{d_3, d_7\}$. $D = \bigcup_{1 \le i \le 4} QDS(q_i)$.

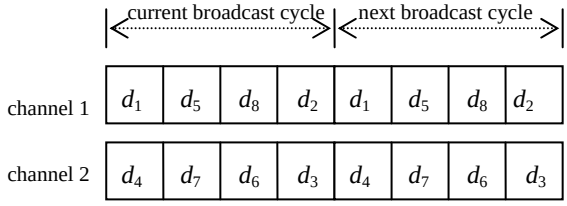

Fig. 2: An example of a broadcast program. There are two channels and eight data items.

Given two data items d_i and d_j, if $\{d_i, d_j\} \subset QDS(q_k)$ and d_i and d_j are scheduled in the same time slot of two different channels. We call that a *data collision* (*DC*) occurred in $QDS(q_k)$, denoted by $d_i \cong d_j$. As shown in example 1, $\{d_1, d_4\} \subset QDS(q_1)$, and d_1 and d_4 are scheduled in the same time slot of channel 1 and channel 2, respectively. Therefore, $d_1 \cong d_4$, and there is a *DC* in $QDS(q_1)$. Since a mobile client can only listen to one channel at the same time, one of the data items, d_1 or d_4, can be accessed in current broadcast cycle and the other one must be waited until next broadcast cycle. It is thus evident that a *DC* will lead to an increase of a client's access time.

Theorem 1: Given a broadcast scheduling and two queries q_i and q_j, if query q_i has a *DS* in its *QDS* and query q_j's *QDS* is not occurred any *DS*, then the q_i's access time is longer than the q_j's access time.

Proof: Figure 3 shows a broadcast scheduling. If $QDS(q_i)=\{d_{i1}, d_{i2}, d_{i3}, d_{i4}, d_{i5}\}$, and $d_{i2} \cong d_{i5}$. The shaded box represents a complete cycle time. (The length of the shaded box is equal to L.) The left

border of the box represents the start time of a client that submits the query q_i to access data items. However we slide the box to anywhere (a client start turning on a channel anywhere), the box will cover all data items one time exactly. The data items which do not induce a *DC* can be accessed during the box (time=L), i.e. d_{i1}, d_{i3}, and d_{i4} can be accessed. But $d_{i2} \cong d_{i5}$, therefore, the client will select one of channels to receipt d_{i2} or d_{i5} at the time. The other data item will receipt until next broadcast cycle. Therefore, the q_i's access time is longer than L. Oppositely, given a query q_j, if all data items in q_j are not inducing any a *DS*. The box will cover all q_j's data items during one length of a cycle time and receipts them. Therefore, the access time of q_j is less than L. So the q_i's access time is longer than the q_j's access time.

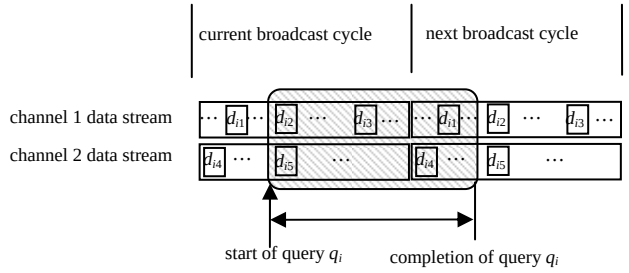

Fig. 3: An example of data broadcast scheduling. It shows the negative influence on access time of a query by a *DC*.

By the theorem 1, hence, this is an important issue to reduce the number of *DC*s in a multi-channel data broadcast program.

Data Broadcast Collision Problem (DBCP): *Given a set of query patterns $Q = \{q_1, q_2, \ldots, q_M\}$ with their frequencies (weights), the data broadcast collision problem is to find an optimal broadcast schedule which minimizes total data collisions of the query patterns.*

3. Our Approach

In this section, we present an algorithm for *DBCP*. Our scheduling algorithm consists of two stages: the partitioning followed by the matching.

3.1 Partitioning Algorithm for the *DBCP*

To obtain an efficient initiation, we use a partition algorithm that partitions all data items into K channels. The objective is to reduce the data collision probability. The main idea is described as follows: if two data items d_i and d_j belong to the same *QDS*, we will partition d_i and d_j into the same broadcast channel as far as possible. Then the two data items will not induce a *DC*. Otherwise, if d_i and d_j are partitioned into different channels, maybe d_i and d_j will be scheduled in the same time slot and inducing a *DC*. We called this is a *tended data collision* (*TDC*). When two data items occur to a *TDC*, they will possible be a *DC* in a data broadcast scheduling. In other words, if two data items belong to the same *QDS* and they are not a *TDC*, then they do not lend to a *DC*.

Given a set of weighted queries $Q = \{q_1, q_2, \ldots, q_M\}$ whose union of *QDS*s is $D = \{d_1, d_2, \ldots, d_N\}$, it can be modeled by a graph $G = (V, E)$, called *query graph*, where V represents the data items and the edge set E represents which two data items belong to the same query. $E = \{e_{i,j} \mid d_i \in QDS(q_k)$ and $d_j \in QDS(q_k)$, where $q_k \in Q\}$. Let $w_{i,j}$ denote the weight of edge $e_{i,j}$. $w_{i,j} = \sum freq(q_k)$, if $\{d_i, d_j\} \subset QDS(q_k)$. For example, given a database D containing six data items $d_1, d_2, \ldots, d_6$, if there are three queries q_1, q_2, and q_3. Assume that $QDS(q_1) = \{d_1, d_2, d_5\}$, $QDS(q_2) = \{d_1, d_5, d_6\}$, $QDS(q_3) = \{d_2, d_3, d_4, d_5\}$, and $ferq(q_1) = ferq(q_2) = ferq(q_3) = 1$. We can represent it by the graph $G = (V, E)$ as shown in the figure 4. $V = \{v_1, v_2, \ldots, v_6\}$ and $E = \{e_{1,2}, e_{1,5}, e_{1,6}, e_{2,3}, e_{2,4}, e_{2,5}, e_{3,4}, e_{3,5}, e_{4,5}, e_{5,6}\}$. Because of $\{d_1, d_5\} \subset QDS(q_1)$ and $\{d_1, d_5\} \subset QDS(q_2)$, $w_{1,5} = ferq(q_1) + ferq(q_2) = 2$. Similarly,

$w_{2,5}$ = *ferq*(q_2) + *ferq*(q_3) = 2 because of {d_1, d_5} ⊂ *QDS*(q_2) and {d_1, d_5} ⊂ *QDS*(q_3). The weights of the other edges are equal to 1.

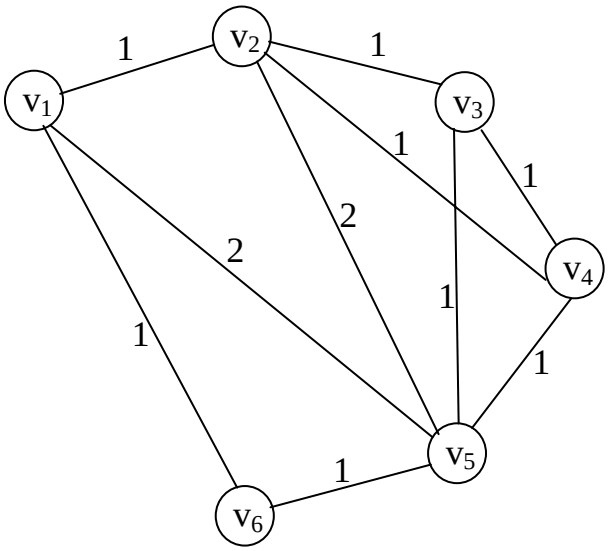

Fig. 4: Graph modeling for weighted query set.

Let |*V*| denote the number of vertices in *V*. The partitioning problem is dividing *V* into *K* sets (*K* channels) V_1, V_2, …, V_K, where

$V_i \cap V_j = \varnothing$, $i \neq j$

$| V_i | = | V_j |$, $1 \le i, j \le K$

$$\bigcup_{i=1}^{K} V_i = V$$

Partition is also referred to as a *cut*. The cost of partition is called the *cut-size*, which is the total weight of edges crossing the cut. A cut-size represents the number of *TDC*s among queries. Let $C_{i,j}$ be the cut-size between partitions V_i and V_j. The objective of the partitioning problem minimizes $\sum_{i=1}^{K}\sum_{j=1}^{K} C_{i,j}$, where $i \neq j$.

Multiway partitioning can be normally reduced to a series of two-way or *bipartitioning* problem. Each component is hierarchically bipartitioned until the desired number of components is achieved. In this paper, we will restrict ourselves to bipartitioning. We call the partitioning process is bisectioning and the partitions are bisections.

Based on KL algorithm [12], we propose a bipartition algorithm in the following. In this bipartitioning algorithm, we start by initially partitioning the graph $G = (V, E)$ into two subsets of equal sizes. Vertex pairs are exchanged across the bisection if exchange improves the cutsize. The above procedure is carried out iteratively until no further improvement can be achieved.

Consider the query graph given in figure 4. The initial partitions are $V_1 = \{v_1, v_2, v_3\}$ and $V_2 = \{v_4, v_5, v_6\}$. (It is shown in figure 5(a).) Notice that the initial cutsize is 8. The next step of the algorithm is to choose a pair of vertices whose exchange results in the largest decrease of the cutsize or results in the smallest increase, if no decrease is possible. The cost reduction for moving vertex v_i, denoted by D_i. D_i is defined as

$D_i = E_i - I_i$,

where E_i is the total cost of edges of vertex v_i that cross the bisection boundary (if $v_i \in V_1$, $E_i = \sum_{v_x \in V_2} w_{x,i}$) and I_i is the total cost of edges of vertex v_i that do not cross the boundary (if $v_i \in V_1$, $I_i = \sum_{v_x \in V_1} w_{x,i}$). If v_i and v_j are exchanged, the decrease of cost is $G_{v_i,v_j} = D_i + D_j - 2w_{i,j}$. In the example given in figure 5(a), for vertex v_1, $E_1 = w_{1,5} + w_{1,6} = 2+1 = 3$, $I_1 = w_{1,2} = 1$, $D_1 = 3-1 = 2$. Similarly, for vertex v_4, $E_4 = 1+1 = 2$, $I_4 = 1$, $D_4 = 2-1 = 1$. In this example, this is a suitable vertex pair, (v_1, v_4), which decreases the cutsize by 3 ($G_{v_1,v_4} = D_1 + D_4 - 2w_{1,4} = 2+1-0 = 3$). A tentative exchange of this pair made. Figure 5(b) shows the result of the tentative exchange. Let g_i denote the decrease of *i*th tentative exchange. Therefore, g_1 is equal to 3. These two vertices are then locked. (The locked vertices represented by shaded nodes.) This lock on the vertices prohibits them from taking part in any further tentative exchanges. If $v_i \in V_1$ and $v_j \in V_2$ are interchanged, then the new *D*-values, *D'*, are given by

$D_x' = D_x + 2w_{x,i} - 2w_{x,j}, \forall v_x \in V_1 - \{v_i\}$

$D_y' = D_y + 2w_{y,j} - 2w_{x,i}, \forall v_y \in V_2 - \{v_j\}$

The above procedure is applied to the new partitions, which gives a second vertex pair of (v_2, v_6). This procedure is continued until all vertices are then locked. Figure 5(c) and 5(d) shows the results of the second and third tentative exchanges, respectively. During this process, a log of all tentative exchanges and the resulting cutsizes is stored. Table 1 shows the log of vertex exchanges for the given example. Let ps_k denote the partial sum of cutsize decrease over the exchanges of first k vertex pairs, $ps_k = \sum_{i=1}^{k} g_i$. Note that the ps-values are given in the column 4 of the table, e.g., ps_1 =3, ps_2 =2 and ps_3 =0. The value of k for which ps_k gives the maximum value of all partial sum is determined from the table. In this example, k=1 and ps_1 =3 is the maximum partial sum. The first k pairs of vertices are actually exchanged. In this example, the first vertex pair (v_1, v_4) is actually exchanged, resulting in the bisection shown in figure 6. This completes an iteration and a new iteration starts. However, if no decrease of cutsize is possible during an iteration, the algorithm stops. Table 2 shows the log of the second iteration. The maximum partial sum is equal to 0. Therefore, no vertex is exchanged and the bisection in figure 6 is the final bisection. The algorithm is summarized in figure 7.

i	Vertex Pair	g_i	ps_i	Cutsize
0	-	-	-	8
1	(v_1, v_4)	3	3	5
2	(v_2, v_6)	-1	2	6
3	(v_3, v_5)	-2	0	8

Table 1: The log of the vertex exchanges. (Iteration 1.)

i	Vertex Pair	g_i	ps_i	Cutsize
0	-	-	-	5
1	(v_2, v_6)	-1	-1	6
2	(v_1, v_4)	-1	-2	7
3	(v_3, v_5)	2	0	5

Table 2: The log of the vertex exchanges. (Iteration 2.)

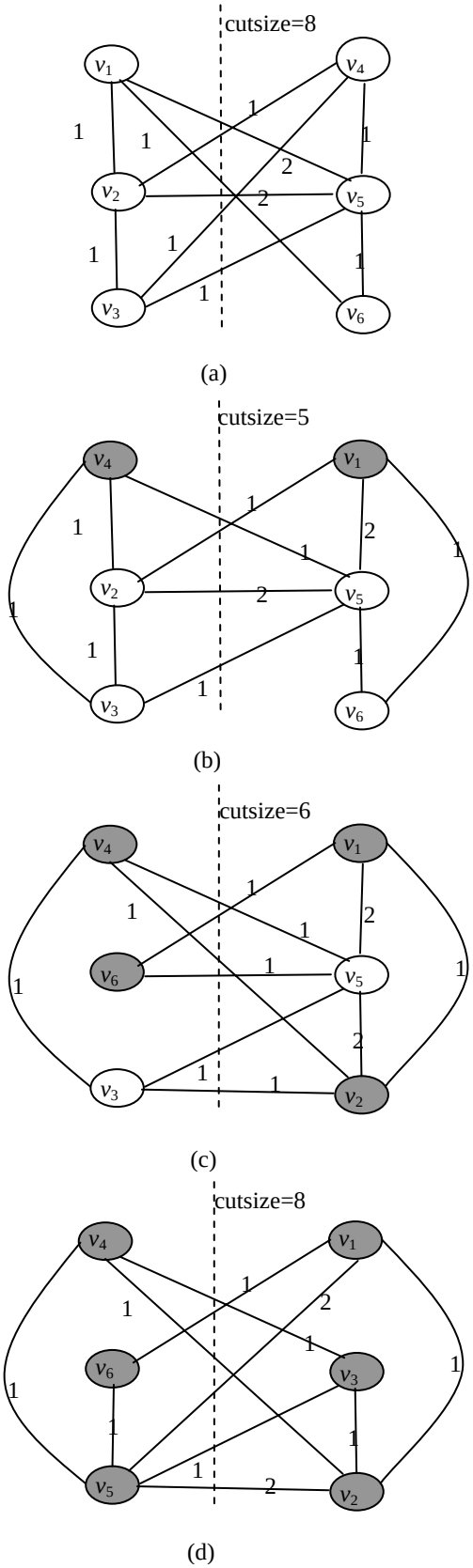

Fig. 5: (a) The initial bisection. (b) The result of the first tentative exchange which exchange vertices v_1 and v_4. (c) The result of the second tentative exchange. (d) The result of the third tentative

exchange.

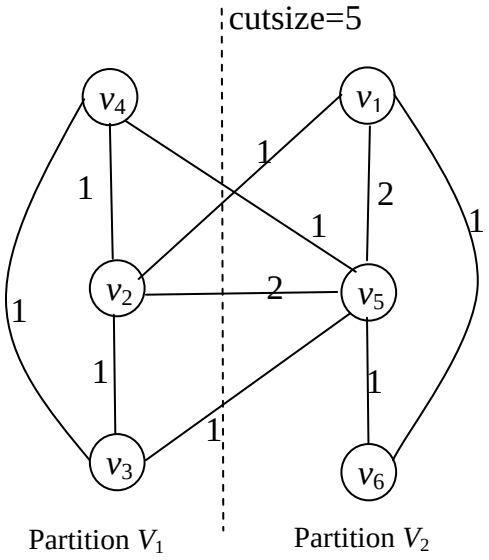

Fig. 6: The final partitions of figure 5.

Algorithm Partitioning (G)

Input: $G(V,E)$, $|V|=2n$

Output: Balanced bi-partition V_1 and V_2 with small number of *TDC*s

1 Random bipartition V into V_1 and V_2 such that $|V_1|=|V_2|$, $V_1 \cap V_2=\varnothing$,and $V_1 \cup V_2$=V;

2 **repeat**

3 Compute D_j, $\forall v_j \in V$;

4 **For** i = 1 **to** n

5 Find a pair of unlocked vertices $v_{x_i} \in V_1$ and $v_{y_i} \in V_2$ whose exchange with largest G_{x_i, y_i};

6 Exchange and mark v_{x_i} and v_{y_i} as locked, store the gain g_i;

7 Compute the new D_j, for all unlock $v_j \in V$;

8 Find k, such that $ps_k = \sum_{i=1}^{k} g_i$ is maximized;

9 **if** $ps_k > 0$ **then**

10 Move v_{x_1}, v_{x_2}, …, v_{x_k}, from V_1 to V_2, and v_{y_1}, v_{y_2}, …, v_{y_k}, from V_2 to V_1;

11 Unlock v, $\forall v \in V$;

12 **Until** $ps_k \leq 0$.

Fig. 7: The partition algorithm for *DBCP*.

3.2 Matching Algorithm for the *DBCP*

Following the partitioning, a matching based method is applied to decide which two data items (partitioned in different channels) will be broadcasted in the same time slot. The matching algorithm has to minimize the number of data collisions. As the example shown in figure 2, it has given us a picture of how the mismatching of data items d_1 and d_4 on the broadcast sequences could increase the access time of the client. In this subsection, we applied the maximum bipartite matching technique to reduce the number of *DC*s.

The maximum bipartite matching technique [6] is a well known method that derived from the Ford-Fulkerson method to find the maximum flow in a flow network. In order to transform the *DBCP* to the maximum bipartite matching problem, we consider the output of the partition algorithm as a bipartite network. Given a graph $G = (V, E)$ that is output from the partition algorithm, where $V= V_1 \cup V_2$ and $V_1 \cap V_2=\varnothing$. We construct the corresponding flow network $G' = (V', E')$ for the bipartite graph G as follows. We let the source s and sink t be new vertices not in V, and we let $V'= V \cup \{s, t\}$. The directed edges of G' are generated by the following rules:

R1: If there are two vertices $v_i \in V_1$, $v_j \in V_2$ and $e_{i,j} \notin E$, then we add an edge $e_{i,j}$ into E' with unit capacity.

R2: $\forall$ $v_i \in V_1$, we add an edge $e_{s,i}$ into E' with unit capacity.

R3: $\forall$ $v_j \in V_2$, we add an edge $e_{j,t}$ into E' with unit capacity.

All edges have only one capacity. This made sure that one unit capacity were shipped from a vertex $v_i \in V_1$ to a vertex $v_j \in V_2$ at most. In other words, a vertex $v_i \in V_1$ is matched to a vertex $v_j \in V_2$ at most. Similarly, a vertex $v_j \in V_2$ is matched to a vertex $v_i \in V_1$ at most. Rule R1 creates en edge between two vertices v_i and v_j if they do not induce a *TDC*, and gives unit capacity to the edge. Therefore, in the flow-based algorithm, v_i and v_j can be matched together and have no *DC* occurring. That is to say, if v_i and v_j inducing to a *TDC*, rule R1 avoids the two vertices matching together and inducing a *DC*. As shown in figure 8(a), this is the output of figure 5, vertices are partitioned into two sets $V_1 = \{v_2, v_3, v_4\}$ and $V_2 = \{v_1, v_5, v_6\}$. There are five *TDC*s between V_1 and V_2. Figure 8(b) shows

the flow network which is corresponding to the bipartite graph of figure 8(a).

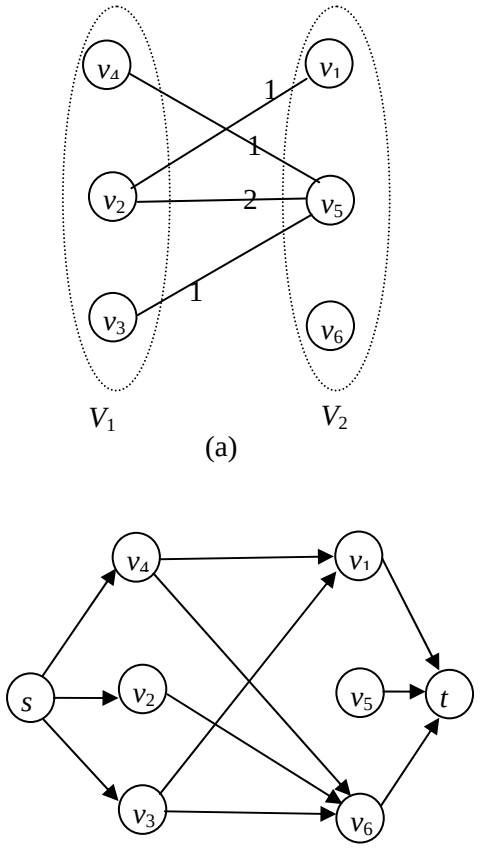

Fig. 8 (a) Bipartite graph $G = (V, E)$, where V is partitioned into two sets V_1 and V_2. (b) The corresponding flow network $G' = (V', E')$.

Apply the Ford-Fulkerson method [6] on the flow network G' to find a maximum flow. We can find a maximum bipartite matching of the graph. As shown in Figure 9, the corresponding flow network G' with a maximum flow is shown. Shaded edges have a flow of 1 unit capacity, and all other edges carry no flow. The shaded edges from V_1 to V_2 correspond to those in a maximum matching of the data items. The vertices v_1 and v_4 are matched together and they will be scheduled in the same time slot of different channels. Similarly, vertices v_2 and v_6 are scheduled in the same time slot. The other vertices v_3 and v_5 are unmatched. Let α denote the set of all matched vertex pairs. In this case, $\alpha = \{(v_1, v_4), (v_2, v_6)\}$.

For unmatched vertices, like v_3 and v_5, we propose a greedy scheme to minimize the number of *DC*s in the following. If there are two unmatched vertices $v_i \in V_1$ and $v_j \in V_2$, that is, there doesn't exist an edge connected v_i and v_j in the flow network G'. In other words, there exists an edge connected v_i and v_j in the graph G. If they are matched together, it will induce $w_{i,j}$ *DC*s. Let V_1'' (V_2'') denote the unmatched vertices in V_1 (V_2), and E'' denote the edges in the query graph G which are between V_1'' and V_2''. In our method, first, we sort the edges in E'' by their weights. Next, we select an edge by the ordering recursively. We select the first edge $e_{i,j}$ with the smallest weight, and match vertices v_i, and v_j together, and add (v_i, v_j) into α. Then, we remove all edges connected with v_i or v_j from E''. The scheme is done to the set E'' emptied. The matching algorithm is summarized in figure 10.

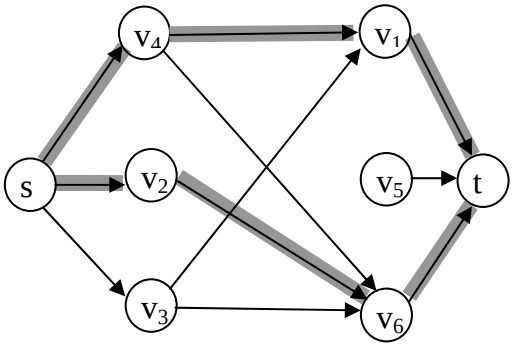

Fig. 9: The corresponding flow network G' with a maximum flow. v_4 will be matched to v_1, and v_2 will be matched to v_6.

Algorithm Matching (G)

Input: $G(V,E)$, $|V|=2n$, V was partitioned into two sets V_1 and V_2 such that $|V_1|=|V_2|$, $V_1 \cap V_2=\varnothing$, and $V_1 \cup V_2=V$

Output: Bipartite matching α of V_1 and V_2 with less number of *DC*s

1 Using rules R1, R2, and R3 to construct the flow network G' by graph G

2 Run the Ford-Fulkerson algorithm on G' to find maximum bipartite matching α

3 **For** i = 1 to $|E|$

4 **if** ($e_{i,j} \in E$, $v_i \in V_1$, $v_j \in V_2$, and $(v_i, v_j) \notin \alpha$) **then**

5 Add $e_{i,j}$ into E'';

6 Sorting the edges of E'' by their weights;

7 **While** ($E'' \neq \varnothing$)

8 Select the first edge $e_{i,j}$ of E'';

9 $\alpha = \alpha \cup \{(v_i, v_j)\}$;

10 Remove the edges connected with v_i,or

v_j from E'';

11 Return α

Fig. 10: The matching algorithm for *DBCP*.

4. Experimental Results

Our algorithms have been implemented in the JAVA language on a PC with a Pentium IV 3.2 G microprocessor and 512 MB RAM. We generate query sets to evaluate the performance of the approach over a range of data characteristics. The parameters used in the generation of query sets include *N*, *M*, *S*, and *F*. Parameter *N* is defined as the number of data items that are delivered on broadcasting channel. Every data item on the broadcast channels is accessed by one or more queries. Parameter *M* is defined as the total number of query patterns that access parts of the broadcast data set. Selectivity *S* is the maximum degree of a query's *QDS* size over the size of broadcast data set in terms of percentage. For example 3% selectivity means a query accesses at most 3% of the *N* data items. We consider three different kinds of distribution of query's occurrence frequency: uniform distribution, normal distribution, and exponential distribution. We assume the size of each data item to be equal and set all data items' size as one unit length. The number of broadcast channels set to 2.

In the first experiment, we change the number of data items *N* with 200 queries and 5% selectivity. The three kinds of distribution of query's occurrence frequency (uniform distribution, normal distribution, and exponential distribution) are considered. The results are shown in Figure 11. We measure the performance improvement i.e., *DC* (Data Collision) reduction of the proposed method against sequential (with respect to data ID) schedules on which no effort of data placement is put. Figure 11 shows the percentages of improvement s of ours over *sequential*. The improvement for the *sequential* is calculated by

$$\frac{sequential - Ours}{sequential} \times 100\%.$$

As show in this result, the performance of the proposed approach has little dependency on the number of data items. The performance improvement increases to more than 30%--40%. The results show the effectiveness of our approach.

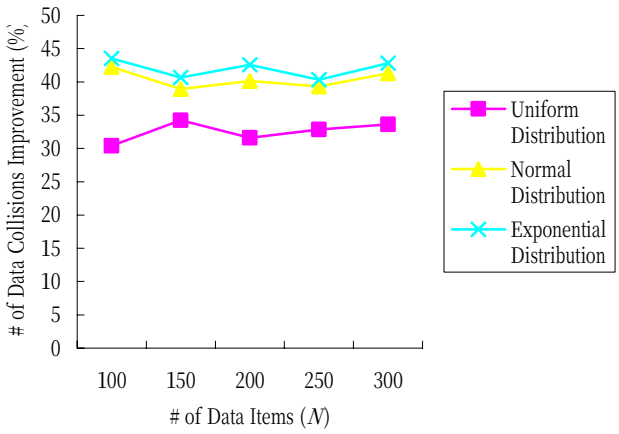

Fig. 11: The improvement with change in the number of data items *N*. Three kind of distribution of query's occurrence frequency, uniform distribution, normal distribution, and exponential distribution, are considered.

In the second experiment, we change the number of query patterns *M* with 300 data items and 5% selectivity. The results are shown in the figure 12. As shown in this result, the performance improvement decreases with large number of queries.

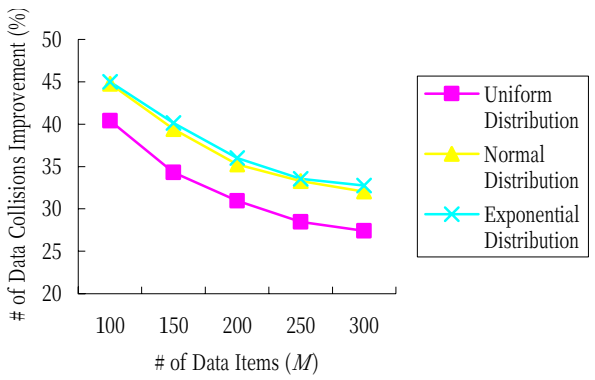

Fig. 12: The improvement ratio with various numbers of query patterns *M*. Three kind of distribution of query's occurrence frequency, uniform distribution, normal distribution, and exponential distribution, are considered.

In the third experiment, we change selectivity values *S* with 100 query patterns and 300 data items. The results are shown in the figure 13. As shown in these results, our proposed method on the average reduces the number of *DC*s occurred by 40.74%, 44.88%, and 46.00%) regarding the uniform distribution, normal distribution, and exponential distribution, respectively.

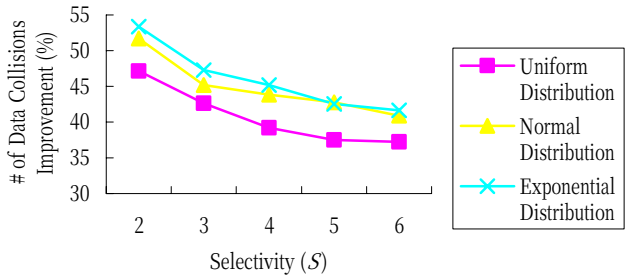

Fig. 13: The improvement ratio with various selectivity *S*. Three kind of distribution of query's occurrence frequency, uniform distribution, normal distribution, and exponential distribution, are considered.

In above experiments we can observe that the cases of exponential distribution of $freq(q_i)$ provide better performance than the others. The results show that more highly skewed query distributions achieve better performance with our method.

5. Conclusions

In this paper, we have formulated a new data broadcast collision problem in multi-channel mobile environment, and we have presented a two-stage algorithm for the problem. The partitioning algorithm decides which data items can be scheduled into the same channel and reduces the data collision probability. Following the partition method, we proposed a flow based matching method to decide which two data items in different channels will be broadcasted in the same time slot. The matching algorithm can minimize the number of data collisions. From our simulation results, we have shown that our approach is efficiently.

References:

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, ``Broadcast Disk: Data Management for Asymmetric Communication Environments," *ACM SIGMOD Conference*, pp. 199--210, 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik, ``Disseminating Updates on Broadcast Disks," *Very Large Data Bases Conference*, pp. 354--365, 1996.

[3] Y.I. Chang and W.H. Hsieh, ``An Efficient Scheduling Method for Query-Set-based Broadcasting in Mobile Environment," *IEEE International Conference on Distributed Computing Systems Workshops*, pp. 478--483, 2004.

[4] Y.I. Chang and C.N. Yang, ``A Complementary Approach to Data Broadcasting in Mobile Information Systems," *Data and Knowledge Engineering*, Vol. 40, pp. 181--194, 2002.

[5] Ming-Syan Chen, Kun-Lung Wu, and Philip S. Yu," Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing", *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 1, 2003.

[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, Second Edition, the MIT Press, 2001, pp. 651--669.

[7] Yon Dohn Chung and Myoung Ho Kim, "An Index Replication Scheme for Wireless Data Broadcasting," *Journal of Systems and Software*, vol. 51, no. 3, 2000.

[8] Y.D. Chung and M.H. Kim, ``Effective Data Placement for Wireless Broadcast," *Distributed and Parallel Databases*, no. 9, pp. 133--150, 2001.

[9] S. Hameed and N. Vaidya, "Efficient Algorithm for Scheduling Data Broadcast," *Wireless Networks*, Vol. 5, No. 3, pp. 183--193, 1999.

[10] T. Imielinski, S. Viswanathan and B. R, Badrinath, "Energy Efficient Indexing on Air," *SiGMOD ACM*, 1994.

[11] T. Imielinski, S. Viswanathan, and B.R. Badrinath, ``Data on air: Organization and access," *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 3, pp. 353--372, 1997.

[12] W. Kernighan, and S. Lin, ``An Efficient Heuristic Procedure for Partitioning Graph," *Bell System Technical Journal*, vol. 49, pp. 291--307, 1970.

[13] Kam-Yiu Lam, Edward Chan, and Joe Chun-Hung Yuen, "Approaches for Broadcasting Temporal Data in Mobile Computing Systems," *The Journal of Systems and Software 51*, 2000.

[14] V.C.S. Lee, K. W. Lam, S. Wu and E. Chan, ``Broadcasting Consistent Data in Mobile Computing Environments," *Proc. Of the 7th IEEE Symp. On Real-Time Technology and Application*, pp. 123--124, 2001.

[15] G. Lee and S.C. Lo, ``Broadcast Data Allocation for Efficient Access of Multiple Data Items in Mobile Environments," *ACM/Baltzer Mobile Networks and Applications*, vol. 8, pp. 365--375, 2003.

[16] Guanling Lee, Shou-Chih Lo, and Arbee L.P. Chen, "Data Allocation on Wireless Broadcast Channels for Efficient Query Processing," *IEEE Transactions on Computers*, vol. 51, no. 10, 2002.

[17] Shou-Chih Lo and Arbee L.P. Chen, "Optimal Index and Data Allocation in Multiple Broadcast Channels," *Data Engineering*, 2000.

[18] W. C. Peng and M. S. Chen, ``Dynamic Generation of data Broadcasting Programs for a Broadcast Disk Array in a Mobile Computing Environment," *Proc. of the 9th International Conference on Information Knowledge Management*, pp. 38--45, 2000.

[19] K. Prabhakara, K.A. Hua, and J.H. Oh, ``Multi-Level Multi-Channel Air Cache Designs for Broadcasting in a Mobile Environment," *Proc. of the 16th International Conference Data Engineering*, pp. 167--186, Feb.-Mar., 2000.

[20] C. Su, L. Tassiulas, and V.J. Tsotras, ``Broadcast Scheduling for Information Distribution," *Wireless Networks*, vol. 5, no. 2, pp. 137--147, 1998.

[21] Kian-Lee Tan and Jeffrey Xu Yu, "Generating Broadcast Programs That Support Range Queries," *IEEE Transactions on Knowledge and Data Engineering*, 1998.

[22] D.A. Tran, K.A. Hua, and K. Prabhakaran, "On the Efficient Use of Multiple Physical Channel Air Cache," *Proc. IEEE Wireless Communication and Network Conference*, pp 17--21, 2002.

[23] Guang-Ming Wu, "An Efficient Data

Placement for Query-Set-Based Broadcasting in Mobile Environment," *Computer Communications*, 2007.

[24] Nitin H. Vaidya and Sohail Hameed, "Scheduling Data Broadcasting in Asymmetric Communication Environments," *Kluwer Academic Publishers*, vol. 5, 1999.

[25] Wai Gen Yee, Student Member, IEEE, Shamkant B. Navathe, Edward Omiecinski, and Christopher Jermaine, "Efficient Data Allocation over Multiple Channels at Broadcast Servers," *IEEE Transactions on Computers*, vol. 51, 2002.