Compression-based Broadcast Data for Reducing Access Time in Wireless Environment

Chao-Chun Chen
Department of Computer Science and Information Engineering,
National Cheng-Kung University,
Tainan, Taiwan, R.O.C.
ccchen@dbserver.csie.ncku.edu.tw

Abstract

On account of the bandwidth limitation of the physical communication medium, the wireless environment is asymmetric communication environment which characteristic of asymmetric communication environment is the download stream capacity from servers to clients is much greater than the upstream capacity from clients back to servers. Traditional client-server protocol to delivery data from servers to clients is not suitable in wireless environment. A better way to delivery data is broadcast-based methods. However, broadcast-based methods usually cause long access time because retrieving broadcast data can be viewed as accessing data from the sequential access device. We proposed two compression algorithms to improve the long access time in this papers. Our proposed algorithms compress the broadcast data pages before they are broadcast, and compressed pages are uncompressed in mobile clients' (MCs') local memory. Due to the limited bandwidth in wireless environment, compression algorithms are suitable to apply in such environment.

keywords: mobile computing, wireless communication, broadcast, compression, mobile database.

1 Introduction

1.1 Asymmetric Communication Environments

In many existing applications the *downstream* communication capacity from server to clients is much greater than the *upstream* communication capacity from clients back to servers. A good example emerges in wireless mobile network. Servers may have a relative high bandwidth broadcast capability, while clients cannot transmit or can do so only over a lower bandwidth cellular link. Systems with such characteristic have been applied in many application domains, including traffic information systems, hospital information systems, public safety applications, and wireless classrooms (e.g. [5, 8]). Such environments are referred as *Asymmetric Communications Environments.*

Communications asymmetry can be caused in two ways: the first is from the bandwidth limitations of the physical communications medium. For example, in wireless networks, servers usually have powerful broadcast ability while mobile clients (MCs) have limited or no such ability. The second is from the patterns of information flow in the application. For example, an information retrieval system in which the number of client is far greater than the number of servers is asymmetric because there is insufficient capacity (either in the network or at the servers) to handle the simultaneous requests generated by the multiple clients.

1.2 Data Delivery in Wireless Environment

In the mobile wireless computing environment, massive number of mobile users will access database over wireless information channels. The access protocol in wireless environment is quite different from the traditional client-server environment. In the traditional client-server information systems, the clients received the required data after sending a request to the server. Such systems are referred as *pull-based.* Pull-based systems is not suitable in asymmetric communication environment on account of a small amount of upstream communication capabilities. This restriction makes only the limited clients be served in the pull-based systems in wireless environment. To ease off the problem, some tasks [6, 7] proposed the broadcast-based data delivery in the wireless environment. In the wireless broadcast environment, systems offer a *broadcast channel* (will be discussed later), and broadcast a great deal of data which is frequently used by MCs in the broadcast channel. Any MC can retrieve data pages from the broadcast channel when the MC needs the external

information. A MC received the required data by sending a request to server only in one situation the required data is not broadcast in broadcast channel. By the broadcast-based data delivery protocol, the number of clients served by the server can be almost infinite scaled. Such systems are referred as *push-based.* It means data is pushed from the server out to the clients.

Data can be viewed as a sequence of pages in the wireless broadcast environment. While a MC needs data from servers, the MC would switch to the broadcast channel and keep on monitoring this channel. When a server broadcasts the desired pages, the MC would store them and quit monitoring the broadcast channel. Therefore retrieving data pages from the broadcast channel can be viewed as sequentially accessing the broadcast program. In practice, servers may broadcast a great amount of data, and it makes much time cost of broadcasting all data large. Sequentially accessing such broadcast data from servers brings long access time for MCs.

One improvement method is the Broadcast Disk method [1]. In general, a *broadcast program* (will be discussed later) generated by the Broadcast Disk method is longer than a flat program, because the Broadcast Disk method will broadcast hot pages more than one time. Hence if a MC retrieve a page which is not hot, access of this page need to spend much time cost.

1.3 Motivation

All broadcast-based methods directly broadcast the data MCs desired over the broadcast channel. In order to reduce access time and without removing data from broadcast data set, we proposed the compression algorithms to archive the goal. Our central idea is compressing the broadcast data pages before generating a broadcast program. In the rest paper, the data pages before the compression process are called *raw pages*; the data pages after the compression process are called *compressed pages.* The broadcast program composed from compressed pages will have a shorter length than other broadcast-based methods. Thus, such a compressed broadcast program will reduce the average access time of MCs. To our best knowledge, no related work focuses on this issue; most past papers made a study of the schedule of broadcast data.

Compressing the raw pages as random order is not efficiency, and two compression algorithms we proposed consider the factor of MCs' requirements. The first compression algorithm compressed the raw pages according to frequency of each pages. In this algorithm, hot raw pages are still in the hot compressed pages; cold raw pages are in the cold compressed pages. Servers can broadcast the hot compressed pages more frequently than the cold compressed pages, then a hot pages can be retrieved soon. The second compression algorithm compressed according to MCs' access patterns. In this algorithm, the average access time will decrease as utility rate of a compressed pages increases. Higher utility rate of a compressed pages can reduce the retrieving times from the broadcast channel, therefore the access time will decrease.

The remainder of this paper is organized as follows. In Section 2, we introduces two related broadcast-based methods and give some assumptions associated with this paper. Our proposed compression algorithms are presented in Section 3. Then a case study of performance is shown in Section 4. Finally, we summarize this paper and describe our future work in Section 5.

2 System Architecture and Related Works

In the study of data delivery in wireless environment, most papers divided the wireless system into two sets of entities: *servers* and *mobile clients (MCs).* In general, servers are powerful stationary machines with database systems, and MCs are portable computing devices, e.g. palmtop or laptop computers. One characteristic of MCs is limited amount of storage space, even diskless. Each MC will retrieve data pages via wireless information channels. The wireless information channels consists of two distinct sets of channels : *uplink* channels and *downlink* channels. The downlink channels consist of *on-demand* channels and a *broadcast* channel. While a MC needs data from servers, he first monitors the broadcast channel. If the MC finds the desired data broadcast from servers, he/she will save it in local memory. Otherwise, the MC sends a request to the server through an uplink channel, and waits for the required data sent by the server in an on-demand channel.

Servers gather statistic in data items used frequently by MCs and broadcast the desired data items in the broadcast channel. The smallest logical unit of the broadcast data is called a *data page* which is made up by data items. The time required to broadcast a data page is referred to as a *time slot.* The server divides the broadcast data into many data pages, and these data pages are referred to as *broadcast set.* Servers schedule a sequence of these data pages to be broadcast. Such a schedule consisted of a sequence of data pages is called a *broadcast program* (or *broadcast cycle*). The server broadcast the broadcast program repeatedly until a new broadcast program is made. After a period of time, MCs will report their access patterns to the servers in order to gather statistic the page access frequency and the relationship of data pages. The server can

generate a broadcast program based on these access patterns.

Some earlier work on broadcast-based database system has been investigated by Datacycle [3, 5]. In their work, the server would simply take the union of the statistic data of MCs and broadcast the resulting set of data pages cyclicly. Each page is broadcast one and the only one time in a broadcast program and such a broadcast program is referred as a *flat broadcast program*. With the flat broadcast, the expected wait for a page on the broadcast is the same for all page (namely, half a broadcast program cycle time) regardless of their relative importance of clients. The flat broadcast is depicted in Figure 1.

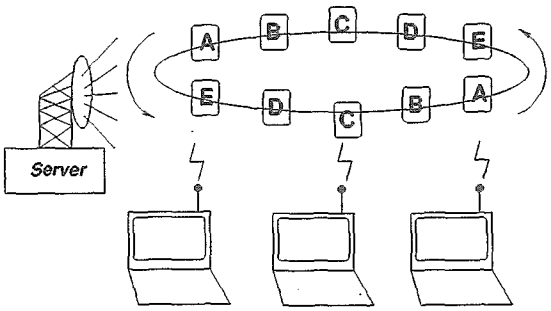

Figure 1: A Flat Broadcast Program

As servers broadcast the program page by page, retrieving data page can be viewed as accessing data from the sequential access device. This makes long access time for MCs. For an example of a flat broadcast program, average access time for retrieving a page is half a broadcast program cycle time. Therefore much related work focused on how to reduce the access time on retrieving data pages in the wireless broadcast environment.

One of the improvement methods is the *Broadcast Disk* method proposed by Acharya et al. [1]. The Broadcast Disk idea is very simple, it assigned the pages to several "air disks" which rotation speed is different. The faster disks are with smaller space while the slower disks are with larger space. Systems would assign the hotter pages to the faster disks, then these hotter pages can be broadcast more frequently. The colder pages would be assigned to slower disk for broadcasting less frequently. An example of a Broadcast Disk algorithm is illustrated in Figure 2. The database (broadcast set) contains 11 pages, and systems assign these pages to three disks, $Disk_1$, $Disk_2$, and $Disk_3$. In this example, $Disk_1$, $Disk_2$, and $Disk_3$ contain 1, 2, and 8 pages, respectively. Speed of $Disk_1$ is twice as speed of $Disk_2$, and speed of $Disk_2$ is twice as speed of $Disk_3$. Hence, pages in $Disk_1$ are to be broadcast twice as frequently as pages in $Disk_2$, and four times as frequently as pages in $Disk_3$. According to the Broadcast Disk algorithm performance analysis, the access time of most MCs will be decreased. However, the parameters of the Broadcast Disk algorithm (including disk number and relative broadcast frequency) should be set painstakingly, a bad setting may generate an unsuitable broadcast program (which length is much longer than a flat broadcast program). An unsuitable broadcast program might make the average access time longer than a flat broadcast program.

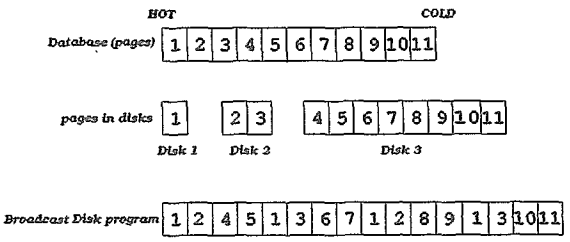

Figure 2: A Broadcast Disk Program

In the past related papers, all of them assumed the pages retrieved by MCs are independent, but such assumption is unreasonable. In the wireless broadcast environment, MCs often need external data to make their applications work successful. When the external data is needed by MCs from servers, the applications will be blocked until the desired data is obtained. Therefore, the time delay applications go on running is depending on the time the desired data is obtained. The data desired by MCs may be in several pages instead of one pages. Thus blocking time of MCs applications is access time to retrieve all these desired pages. Only discuss access time retrieving one page is less meaning because the MCs application still need to wait for other desired pages to go on running. For example, a client application, C_i needs P_1, P_2, and P_3 three pages. When C_i obtains P_1, C_i is still monitoring the broadcast channel because it can't continue executing without P_2 and P_3. Only after obtaining all P_1, P_2, and P_3 pages, C_i can go on executing. Hence, computing access time that C_i retrieve all desired pages is more meaningful than access time to retrieve individual page. It is hard to know the number of pages a client application needed beforehand, so we consider a requirement of a MC needs to retrieve data from the broadcast channel as a *query*. A query contains one or more pages, it depends on the number of data items satisfied the query. Our interesting performance metrics is computing response time to access all pages satisfied a query, instead of response time accessing an individual page.

This paper focus on wireless broadcast environment. Some assumptions should be restricted in order to make our initial work feasible. These assumptions include:

- The server broadcasts page over a single channel. All MCs retrieve data pages from this single channel.

- When a MC switch to the public channel, it can retrieve data pages immediately. The delay for hardware/software preparation to begin monitoring the broadcast channel is short. The short delay doesn't make a great effect on our work, hence, we assume a MC can retrieve data pages immediately when it switch to the broadcast channel.

- MCs retrieve data pages from the broadcast when external data is needed; there is no prefetching.

- We assume the MCs are simple and without a great amount of memory; there is no cache scheme on the MCs.

3 Compressed Broadcast Data Algorithms

It is effective to delivery data by broadcast protocol in wireless environment. However, a drawback is that accessing the broadcast data is sequential access. When MCs retrieve data pages, long access delay will be caused. The Broadcast Disk method is proposed to alleviate the problem of long access delay. The Broadcast Disk method uses the access frequency of each page to generate a Broadcast Disk program, and this program makes most MCs retrieve desired data pages in shorter access time than a flat program. However, the length of a Broadcast Disk program is longer than the length of a flat program, access time of some pages (cold pages) will become longer. If some of data pages satisfied a user's query are in the cold pages, access time of the query is costly. Even though most pages are retrieved in short time, a MC might spend much access time on monitoring the broadcast channel just only one of desired page is in cold pages.

To reduce the access time for retrieving pages satisfied a query, a broadcast program which can offer shorter access time than the Broadcast Disk method is necessary. We proposed two *compressed broadcast data algorithms* to reduce the access time for retrieving pages satisfied a query. Our central idea is compressing the data pages before they are broadcast, then generating a broadcast program by using these compressed data pages. An advantage of this kind of compressed broadcast programs is offering shorter access time for retrieving all data pages, because the length of the compressed broadcast programs is shorter than original broadcast programs. A compressed page will be uncompressed in MC's local memory, and few time cost is spent on the uncompressing operation. Usually, size of a compressed page is less than 10K bytes, the time cost of uncompressing a compressed page is much less than time cost of receiving a compressed page from the broadcast channel. Therefore, time cost spent on the uncompress operation can be ignored.

We proposed two compression algorithms to compress broadcast data pages, one is compressing raw pages based on frequency of raw pages; the other is compressing raw pages based on query. The goal of compression algorithms is reducing the length of broadcast programs, and makes the access time of data pages become shorter. The first proposed compression algorithm compresses the pages with similar frequency into compressed pages; and the second proposed compression algorithm compress the pages which appear together in a query into compressed pages. A broadcast program is generated by using these compressing the raw pages, and the Broadcast Disk algorithm is employed to do such the work. Our work focused on proposing algorithms to reducing access time of retrieving data pages over the broadcast channel, instead of how to schedule the broadcast pages. Therefore, we used an existing schedule algorithm in our proposed algorithms, and didn't offer a new schedule algorithm in this research. The reason we adopted the Broadcast Disk algorithm to schedule the compressed pages rather than a flat broadcast method is the former algorithm offers better performance. The analysis of these two methods can be found in [1, 2].

Our proposed compression algorithms can be divided into tow phrases. The first phrase is generating the compressed pages set. This phrase can be done by compressing raw pages based on either frequency or query. The second phrase is generating a broadcast program, and the Broadcast Disk algorithm is employed to finish this work. In the following, we'll elaborate our proposed algorithms.

3.1 Algorithm 1: Compression Based on Page Frequency

Our first proposed compression algorithm is compressing raw pages based on frequency of each raw page, and raw pages with similar frequency will be compressed together. That is, a hot page will be compressed with other hot pages; and a cold pages will be compressed with other cold pages. A characteristic of this compression algorithm is a hot raw page will be still in a hot compressed page; and a cold raw page will be in a cold compressed page. A broadcast program which is generated by these compressed pages and the Broadcast Disk algorithm preserves the property of traditional Broadcast Disk method, i.e., the broadcast times of hot data pages is more than the cold data pages. Another advantage of this algorithm compared with the traditional Broadcast Disk method is this algorithm produces a short broadcast program.

A variable, *compression_ratio* is used to control the number of pages which are to be compressed together. The different compression algorithms are employed, the *compression_ratio* will be different. In first phrase, this algorithm first sorts the raw pages by their frequency from hot to cold. Then every *compression_ratio* pages from hot to cold in the sorted raw pages will be compressed as a set of compressed pages. In the second phrase, we borrowed the Broadcast Disk algorithm to generate compressed broadcast program. Scheduling method can be other one, even a new one to do this work.

The details of algorithm are described as follows.

Algorithm 1: Compression Based on Page Frequency

Require: raw pages, P_i, where $1 \leq i \leq n$. *compression_ratio* (the number of pages which will be compressed together).

Ensure: *CompressedProgram* is the compressed broadcast program.

1: Sort raw pages by their frequency as $P_1, P_2, \cdots, P_n$, where P_1 is the page with the most frequency, and P_n is the page with the least frequency.
2: **for** i = 1 to n step *compression_ratio* **do**
3: $CompressedPage_i.content = compression(P_i.content, P_{i+1}.content, \cdots, P_{i+compression\_ratio-1}.content)$;
4: $CompressedPage_i.freq = \frac{P_i.freq+P_{i+1}.freq+\cdots+P_{i+compression\_ratio-1}.freq}{compression\_ratio}$;
5: **end for**
6: **if** n *mod compression_ratio* $\neq 0$ **then**
7: $CompressedPage_{i+1}.content = compression(P_n, P_{n-1}, \cdots, P_{n-(n \bmod compression\_ratio)+1})$;
8: $CompressedPage_{i+1}.freq = (P_n.freq + P_{n-1}.freq + \cdots + P_{n-(n \bmod compression\_ratio)+1}.freq)/(n \bmod compression\_ratio)$;
9: **end if**
10: $CompressedProgram = BDalgorithm(CompressedPage_1, CompressedPage_2, \cdots, CompressedPage_{\lceil \frac{n}{compression\_ratio} \rceil})$;
11: Algorithm completes. The output is *CompressedProgram*.

Note that two functions, *compression* and *BDalgorithm* are employed in this algorithm. The *compression* function is used to compress the pages in the parameters, and returns the compression result. Designers can use any existing compression algorithm to achieve this object. The *BDalgorithm* function is used to generate a broadcast program. This program generating function is borrowed from the Broadcast Disk algorithm [1]. This function will generate a broadcast program by the Broadcast Disk algorithm, and the parameters are the compressed pages and their associated frequency. As we mentioned in Section 2, Broadcast Disk algorithm needs four parameters to generate a broadcast program. We don't decide the disk number and the relative frequency of each disk, because these two parameters setting should be generated automatically by the *BDalgorithm*.

3.2 Algorithm 2: Compression Based on Query

This algorithm compresses raw pages by relations between the data pages and queries. In order to record the relation between data pages and queries, servers should maintain a table, called *query table*. This table contains two fields, one is *Queries* and the other is *Desired data pages*. The *Queries* field stores query identities which are used frequently by MCs; the *Desire data page* field stores the data pages which are needed in the corresponding query. Servers can know which pages would satisfied a query through the query table. Figure 3 shows the schema of query table. The schema of the query table is quite simple and easy to implement. The size of each record in the table can be as 42 byte wide, if size of a *Queries* value is two bytes, and size of a *Desired data pages* value is 40 bytes (assume a page size is four bytes, and a query contains at most ten pages). Hence, even if a server is managing 100,000 queries and their associated pages (which is a large number for the query table), the size of the query table is only 4.2Mbytes. It is easily handled by any current DBMS. The data of the query table is used to generate a broadcast program, and can be deleted after a new broadcast program is made up. Thus, the size of the query can't grow up, only fixed disk space is needed for the query table.

Queries	Desired data pages

Figure 3: A schema of query table

The second algorithm proposed in this paper is compression based on query. The server gather statistic in the information which queries issued by MCs are used frequently. According to the *Desired data pages* of the queries from query table, pages appeared in the same query are compressed together as far as possible. Compressing raw pages in such way reduce not only the broadcast program length, but also the times of retrieving compressed pages from the broadcast channel. Reducing the times of retrieving compressed pages from the broadcast channel means reducing access time for retrieving total pages of a query.

To represent the relation between data pages and queries, the data structure, *graph* is used to describe such relations. The vertices in graph denote the pages, and the edges denote the relation between pages and queries. Assume v_i and v_j represent P_i and P_j, respectively, the edge, e_{ij} exists if P_i and P_j are in the same query. We also define the weight, w_{ij} on the edge, e_{ij} as the frequency that P_i and

P_j are in different queries. For example, the server has the information about pages and queries as Figure 4. There are three nodes in this graph because the pages union of queries, $\{Q_1, Q_2, Q_3\}$ is $\{P_1, P_2, P_3\}$. From the pages in Q_1, the edge, e_{12} is added to the graph due to P_1 and P_2 are in Q_1. In such manner to examine Q_2 and Q_3, two additional edges, e_{13} and e_{23} are added to the graph. Deciding the weight on a edge needs to scan one pass of query table. w_{12} equals to two because P_1 and P_2 appear together in Q_1 and Q_2. w_{13} and w_{23} can be computed by the similar way. The result of the transformation from the query table of Figure 4 to a graph is depicted in Figure 5.

Queries	Desired data pages
Q_1	P_1, P_2
Q_2	P_1, P_2, P_3
Q_3	P_2, P_3

Figure 4: An example of query table

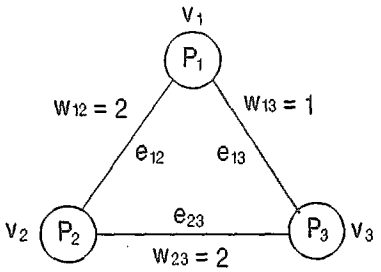

Figure 5: The graph representation of Figure4.

We use a heuristic method to make the pages appeared together in the same query be compressed. By this way, the utility rate of a compressed page retrieved by an MC will increases, that is, more raw pages in a compressed page are the desired pages of an MC. Increasing the utility rate of a compressed page can reduce the times MCs retrieve the broadcast channel. Reducing the times MCs retrieve the broadcast channel also means reducing the access time for retrieving desired pages. In first phrase of this algorithm, our heuristic method first selects the vertices adjacent to the edge with maximum weight as *compression set* (vertices in this compression set will be compress together). If the number of compression set is less than *compressi_ratio*, then find a vertex which is not in the compression set and the weight the vertex connects to the compression set is maximum. After finding such a vertex, add this vertex to the compression set. Continue this step adding vertices to the compression set until the number of compression set equals to *compressi_ratio*, then the pages in the compression set are compressed into a compressed page. The algorithm will compress such pages as possible. But some rest raw pages may be still not compressed in this way, the algorithm will compress them into compressed pages as random order. Give an example of compressing pages in Figure 5, and assume *compressi_ratio* is set to 2. Our heuristic method will find e_{12} is the edge with maximum weight, then v_1, and v_2 are added to the compression set. We mark the compress set with a bold circle in Figure 6. Now, because the number of compression equals to *compressi_ratio*, P_1 and P_2 will be compressed into a compressed page. This step is depicted in Figure 6. After compressing all raw pages, the Broadcast Disk algorithm is employed to generate a broadcast program, and it is the work of the second phrase.

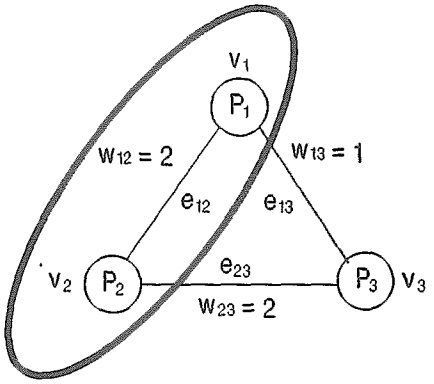

Figure 6: A suitable selection to compress data pages.

We also use compressed pages to generate a broadcast program by the Broadcast Disk algorithm. In order to use the Broadcast Disk algorithm, frequency of each compressed page should be known. A way to take the compressed page frequency is computing the average of frequency of pages in the compressed page.

The details of algorithm are described as follows.

Algorithm 2: Compression Based on Query

Require: raw pages, P_i, where $1 \leq i \leq n$. *compression_ratio* (the number of pages which will be compressed together),

Ensure: *CompressedProgram* is the compressed broadcast program.

1: $S = \emptyset$; *counter* $= 0$;
2: Use a graph to represent the relation between pages and queries. The nodes, v_i and v_j represent the pages, P_i and P_j, respectively. The edge, e_{ij} exists if P_i and P_j are in the same query. The weight of edge, e_{ij}, is the frequency that P_i and P_j are in different queries.
3: **if** $S \neq \emptyset$ **then**
4: Find out the edge with maximum weight, e_{ij}, where $v_i \in S$, and v_j is not marked;
5: **if** e_{ij} is found **then**
6: $S = S \cup \{v_j\}$; {Add v_j to S.}

7: else
8: mark all edges connected to the points in S;
9: $S = \emptyset$;
10: end if
11: else
12: Find out the edge with maximum weight, e_{ij}, where v_i, v_j, and e_{ij} are not marked;
13: $S = \{v_i, v_j\}$;
14: end if
15: Mark v_i, v_j, e_{ij};
16: if $number(S) == compression\_ratio$ then
17: Compress pages in S as $CompressedPage_{counter}.content$; {A *compression* function is employed here.}
18: Compute the average frequency of these pages as $CompressedPage_{counter}.freq$;
19: $counter = counter + 1$;
20: $S = \emptyset$;
21: end if
22: if there exists a node isn't marked then
23: go to step 3;
24: else
25: go to step 27;
26: end if
27: Compress the pages didn't be compressed yet into compressed pages by random order. Assume there are total m compressed pages.
28: $CompressedProgram$= $BDalgorithm(CompressedPage_1, CompressedPage_2, \cdots, CompressedPage_m)$;
29: Algorithm completes. The output is *CompressedProgram*.

Note that two functions, *compression* and *BDalgorithm*, are also employed in this algorithm. The two functions are basically the same as those discussed in last subsection.

4 Performance Discussions: A Case Study

As our experiment models are developing, the complete performance analysis is not demonstrated in this paper. We give a case study to show a rough performance result for broadcast-based methods in this section.

In this case study, 11 pages are in the broadcast set, broadcast programs generated by three broadcast methods will be compared. The first compared program is a flat program. The flat program is composed by these 11 pages. The second compared program is a Broadcast Disk program. This Broadcast Disk program is the same as Figure 2, details of the program can be found in Section 2. The third compared program is our proposed compressed program generated by Algorithm 1. Only one of our proposed compressed algorithms is compared in this case study, because the model of the compressed program generated by Algorithm 2 needs more assumptions, such as query table. We didn't ready complete models yet, hence, only Algorithm 1 is used to compare with other methods. In the third program, two row pages are compressed into one compressed page. The parameter, number of disk, in the Broadcast Disk algorithm used in Algorithm 1 is two, and pages in the first disk broadcast twice as frequently as pages in the second disk. The first disks contains two pages, one is compression of 1 and 2; the other is compression of 3 and 4. Other compressed pages are in the second disk. These three broadcast programs are illustrated in Figure 7.

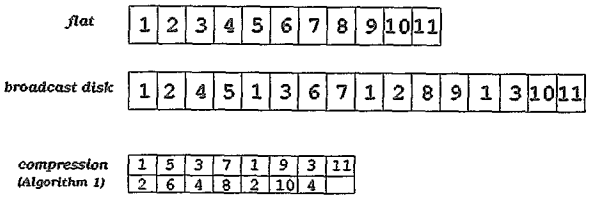

Figure 7: Three compared broadcast programs

Some assumptions is restricted in this case study, including:

- A query result contains only one page.
- Using *time slot* as the unit to calculate the expected delay. 1 *time slot* denotes the time cost that servers broadcast a page.
- The length of each page is fixed. This assumption makes the *time slots* of each page are equal.

Access time is the sum of expected delay and page retrieving delay. Because the length of page is fixed, expected delay of the desired pages is the main factor to determine access time. Hence, calculating access time can be replaced by calculating expected delay. Figure 8 shows the expected delay for page requests given various MC access probability distributions, for the three compared broadcast program. The expected delay is calculated by multiplying the probability of access for each page times the expected delay for that page and summing the results. Note that in our probability distributions, page 5 to page 11 are with the same value of probability, we only use one column to record it for the convenience to read.

From Figure 8, it is clear to observe our proposed compression algorithm is more outstanding than other two methods in most probability distributions. This performance improvement is from the shorter program cycle. Our compressed program length is eight, but the Broadcast Disk program length is 16. Long program length increases the expected delay for a page. Hence, the Broadcast Disk program makes longer access time than our compression algorithms. Furthermore, expected

Access Probability					Expected Delay (in time slots)		
1	2	3	4	5 ~ 11	flat program	Broadcast Disk program	compressed program (Algo. 1)
0.34	0.165	0.165	0.04125	0.04125	5.5	4.64	2.5775
0.50	0.125	0.125	0.03125	0.03125	5.5	4	2.4375
0.75	0.0625	0.0625	0.0015625	0.0015625	5.5	3	2.225
0.90	0.025	0.025	0.00625	0.00625	5.5	2.4	2.0875
1	0	0	0	0	5.5	2	2

Figure 8: Expected delay for various programs

delay of our compressed program is stable no matter the access probability is skew or not (expected delay varied from 2.5775 to 2), however, the Broadcast Disk program is great effected by the data skew factor (expected delay varied from 4.64 to 2). It shows our proposed compression algorithms are suitable in most situations.

5 Conclusions

Broadcast-based methods are used widely in asymmetric communication environments, including wireless environment. However, a drawback is wireless data broadcasting can be viewed as data in the sequential access device on the air. When MCs need external data, they would monitor the broadcast channel and wait for the server broadcast broadcast the desired data. For the sake to reduce access time, we proposed two compression algorithms to compress the raw pages before server broadcasts a program. Our proposed algorithms make the broadcast program become shorter than one generated by Broadcast Disk algorithm or flat broadcast. Hence, the broadcast cycle time of compressed broadcast programs become shorter than program cycle time generated by other methods, and all pages are still broadcast. The shorter broadcast cycle time is beneficial for all mobile clients to reduce access time of any data page.

Our future works will focus on the study of performance analysis of the proposed two algorithms. We will complete the performance experiments, then show the improvement of our two compression algorithms. When an MC retrieves compressed pages from the broadcast channel, many raw pages in the compressed pages are discarded. To improve the utility rate of the raw pages in compressed pages, one possible way is to co-operate with the cache strategy in the MCs. Increasing the utility rate of a compressed page is another work of our next step.

References

[1] Swarup Acharya, Rafael Alonso, Michael Franklin, and Stanley Zdonik, "Broadcast Disks : Data Management for Asymmetric Communication Environments", *Proceedings of the 1995 ACM SIGMOD Conference*, San Jose, California, May 1995, pp. 199-210.

[2] Swarup Acharya, Michael Franklin, and Stanley Zdonik, "Prefetching from a Broadcast Disk", *Proceedings of the 12th International Conference on Data Engineering*, New Orleans, Louisiana, February 1996, pp. 276-285.

[3] T. Bowen, et al. "The Datacycle Architecture", *CACM*, Vol. 35, No. 12, December 1992.

[4] Veena Gondhalekar, Ravi Jain, and John Werth, "Scheduling on Airdisks : Efficient Access to Personalized Information Services via Periodic Wireless Data Broadcast", *IEEE International Conference on Communications(ICC)*, June 1997, pp. 1276-1280.

[5] T. Imieliński and B. Badrinath, "Mobile Wireless Computing: Challenges in Data Management", *CACM*, Vol. 37, No.10, October 1994.

[6] T. Imieliński and S. Viswanathan, "Adaptive Wireless Information Systems", *Proceedings of Special Interest Group in Database Systems(SIGDBS) Conference*, Tokyo, Japan, October 1994.

[7] T. Imieliński, S. Viswanathan, and B. R. Badrinath, "Data on Air : Organization and Access", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, No. 3, May 1997, pp. 353-372.

[8] R. Katz, "Adaption and Mobility in Wireless Information System" *IEEE Personal Comm.* 1st Quarter, 1994.

[9] Chih-Horng Ke and Chiang Lee, "Minimizing Expecting Time in Broadcast-Based Mobile Computing Environments", *Proceedings of 1996 Workshop on Distributed System Technologies and Applications*, May 1996, Tainan, Taiwan, R.O.C. pp. 287-297.