International Symposium on Wireless Pervasive Computing
ISWPC 2006

2006 1st International Symposium on Wireless Pervasive Computing

January 16-18, Phuket, Thailand

IEEE

IEEE COMMUNICATIONS SOCIETY

KMUTT

Siam Photon

ASIAN OFFICE OF AEROSPACE R&D

ECTI-21 NECTEC

Proc. of Int'l Symposium on Wireless Pervasive Computing
(ISWPC 2006), pp. 450-455 (Jan. 2006).

A Query Processing Mechanism Based on the Broadcast Queue for Broadcast Database Systems

Shinya KITAJIMA*, Jing CAI*, Tsutomu TERADA†, Takahiro HARA* and Shojiro NISHIO*
* Dept. of Multimedia Eng., Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita-shi, Osaka 565-0871, Japan
Email: {kitajima.shinya, cai, hara, nishio}@ist.osaka-u.ac.jp
† Cybercommunity Division, Cybermedia Center, Osaka University
5-1 Mihogaoka, Ibaraki-shi, Osaka 567-0047, Japan
Email: tsutomu@cmc.osaka-u.ac.jp

***Abstract*— In recent years, there has been an increasing interest in the broadcast database system, where the server periodically broadcasts contents of a database to mobile clients such as PDAs. There are three query processing methods in the broadcast database system: (i) the server processes a query and then broadcasts the query result to the client; (ii) the client stores all data that are necessary in processing the query and then processes it locally; and (iii) the server and the client collaborate in processing the query. Since the performance of each method changes according to the situation, such as the interval of query generation and the size of query results, it is difficult to decide the optimal method among them statically. In this paper, we propose a new query processing method which dynamically changes the order of queries submitted in the queue at the server and also changes processing methods for the queries according to the deadline of queries and the system situation. Our method not only improves the response time but also increases the success rate of query processing compared with the traditional methods.**

I. Introduction

The recent evolution of wireless communication technologies has led to an increasing interest in information systems in which data is disseminated via the broadcast channels. In such systems, a server broadcasts various data periodically via the broadband channel, while clients pick out and store necessary data. Since the data delivery cost of the server little increases even if the number of clients increases, it can disseminate data in high throughput independently of the number of clients.

There are many studies for improving the performance of broadcast database system, which include scheduling techniques at the server side [1, 5, 7, 8], caching techniques at the client side [1, 8], update propagation techniques [2], integration of push-based broadcast and pull-based broadcast [3], and pre-fetching techniques [4]. Most of them deal with broadcast data as data items simply, and do not address the performance improvement by considering contents and characteristics of broadcast data. Since broadcast systems employ various data formats, such as hypertexts form and relational data according to applications, the data processing mechanism that considers the characteristics of broadcast data becomes important to improve the system performance.

In this paper, we assume the broadcast system that the server broadcasts contents of a relational database and clients issue queries to retrieve data from the database. We call such system the *broadcast database system*. There are three query processing methods in the broadcast database system: the on-demand method, the client method, and the collaborative method. Since the performance of each method changes according to the situation such as the interval of query generation and the size of query results, it is difficult to decide the optimal method among them statically.

In this paper, we propose a new query processing method which chooses the query processing method with the least response time among these three methods. Moreover, we propose a query processing method which dynamically changes the order of queries submitted in the queue at the server and also changes processing methods for the queued queries according to the deadline of queries and the system status.

The remainder of this paper is organized as follows. Section II describes the outline of the broadcast database system. Section III explains our method in detail, and Section IV evaluates the performance of our method. Section V discusses the related work, and we conclude the paper in section VI.

II. Broadcast Database System

Figure 1 illustrates the concept of the broadcast database system. In the system, the server broadcasts contents of a relational database via the broadcast channel and clients issue queries to retrieve the necessary data from the database. We give out the following assumptions:

Server: The server periodically broadcasts contents of a relational database. The server also processes queries from clients.

Clients: Clients have a small storage, low battery, and low CPU power such as a PDA.

Downlink channel: The broadcast channel from the server to clients is divided into two channels: broadband *main channel* to disseminate the contents of a database repeatedly, and narrowband *sub channel* to disseminate the other data.

Uplink channel: There is a narrowband uplink channel from clients to the server. Clients use the uplink channel to send queries to the server.

A. Assumed Environment

We assume that our method is used for disseminating information to many and unspecified users in town. For example,

0-7803-9410-0/06/$20.00 ©2006 IEEE.

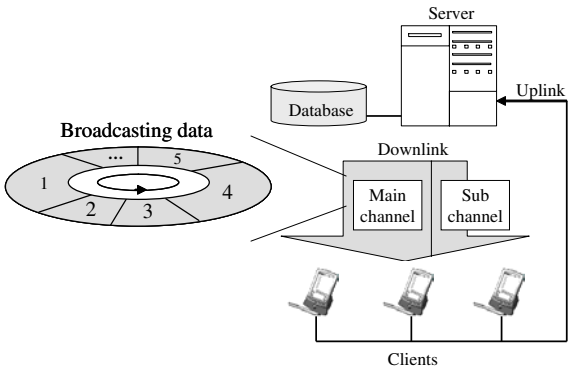

Fig. 1. Broadcast database system.

in the information service for a shopping center, the server broadcasts the database including the advertising information, shop information, and goods information in the shopping center, while hundreds of users receive the broadcast information and retrieve the goods information. Although clients are usually satisfied with receiving broadcast information, clients occasionally issue queries to the server to retrieve the information, such as a natural join operation "I want the image of item *A*, and the map of the shop selling the item". We assume several minutes' delay of receiving the query result is acceptable for the clients. On the other hand, clients can set the deadline to each query. When clients cannot receive the query result in the time of deadline, the query fails.

B. Query Processing Methods

In the broadcast database system, there are three query processing methods as follows.

1) On-demand Method: A client sends a query to the server through the uplink. The server processes the query and broadcasts the query result via the sub channel.

In this method, since the query processing is completely done by the server, no workspace is required for the query processing at client. Additionally, when the query frequency is low, the waiting time of receiving the query result becomes short. However, since the sub channel is exhausted when queries are issued frequently or when the size of query results is large, it takes long time for the clients to receive the query result.

2) Client Method: A client stores all the tables related to the query. Then it processes the query by itself.

In this method, even if the number of clients increases, each client can receive all the necessary data within one broadcast cycle, and then get the query result. Moreover, this method does not require the uplink, therefore it can work even if there is no uplink. However, a large storage is required on the client since the client has to store all the necessary tables for query processing. Moreover, query processing is a heavy workload for the client.

3) Collaborative Method: A client sends a query to the server through the uplink. The server processes the query, attaches the query identifier to the tuples that appear in the query result, creates rules for the client to process the data, and then broadcasts the rules via the sub channel. Based on the received rules, the client receives the necessary tuples from the broadcast database via the main channel referring to the identifiers, and reconstructs the query result automatically by combining these tuples [6].

```
E : RECEIVE
C : Now.minutes>50
    Now.minutes<70
    New.Q_ID = 3
A : STORE( 3, "X", New.A )
```

Fig. 2. An example of ECA rule.

We use the ECA rule to describe processing rules, which is the behavior description language for an active database. Figure 2 shows an example of the ECA rule. In this example, a client listens the broadcast channel between times 50 and 70 and receives the attribute *A* which has identifier 3 of table *X*.

In this method, since a client only needs to store the data necessary to reconstruct the query result by referring to the identifiers, the storage size required on the client is reduced compared with the client method. Moreover, since the size of an ECA rule is generally much smaller than that of a query result, the sub channel is rarely exhausted in this method compared with the on-demand method. However, the response time of the collaborative method is slightly longer than that of the client method, since the broadcast cycle becomes longer when attaching identifiers. Additionally, when queries are issued frequently, the success rate of the query processing becomes lower due to the lack of the identifiers.

III. Selection of Query Processing Method

As mentioned in Section II-B, the system performance, when each method is used individually, changes according to the environmental conditions, such as the query issue frequency and the size of the query result. Thus, the total system performance can be improved if the server can choose an appropriate method among the three query processing methods according to the change of environmental conditions.

In this section, we propose several methods which dynamically choose a query processing method. Firstly, we describe the LRT method which chooses a query processing method simply based on the response time, and then describe the extended methods which are all based on the LRT method.

A. LRT Method

In the LRT (Least Response Time) method, when the server receives a query from a client, it calculates the response time respectively for the on-demand method, the client method, and the collaborative method, and then chooses a query processing method with the least response time.

1) Calculation of Response Time: The system calculates the response time of the on-demand method, the client method, and the collaborative method in the LRT method. The sum of the transmitting data size the up to kth query in the broadcast queue is represented as $S(k)$, the broadcast bandwidth of the sub channel is B_s, and the present time is T_{now}. Since the time for a client to transmit a query to the server, and the time for the server or the client to spend in query processing are very short, compared with the time to broadcast the data, we ignore them.

On-demand method: The response time for the on-demand method T_{on} is the sum of the time to finish transmitting all the data in the broadcast queue of the sub channel and the time to send the query result. When there are n broadcast data in the broadcast queue of the sub channel and the size of query result is s_{on}, T_{on} is calculated as follows:

$$T_{on} = \frac{S(n) + s_{on}}{B_s}.$$

Client method: The response time for the client method T_{cl} is the time until all the required tables are broadcast via the main channel. Here the start time of the nearest broadcasting for the required table t_{cl_s} meets the following condition:

$$t_{cl\_s} \geq T_{now}.$$

T_{cl} is calculated as follows, where t_{cl_e} is the time in which all the required tables will be finished broadcasting within one broadcast cycle:

$$T_{cl} = t_{cl\_e} - T_{now}.$$

Collaborative method: The response time for the collaborative method T_{co} is the sum of the time to finish transmitting all the data before the position where the processing rule is inserted in the broadcast queue of the sub channel, the time to send the processing rule, and the time to finish receiving the required tuple after the client receives the processing rule. The size of the processing rule is represented as s_{co}. Assume that the processing rule is inserted into the mth of the broadcast queue. Here the start time of the nearest broadcasting for the required tuple t_{co_s} meets the following condition:

$$t_{co\_s} \geq T_{now} + \frac{S(m-1) + s_{co}}{B_s}.$$

T_{co} is calculated as follow, where t_{co_e} is the time in which all the required tuples will be finished broadcasting within one broadcast cycle:

$$T_{co} = t_{co\_e} - T_{now}.$$

2) Query Processing Algorithm: The query processing procedure of the LRT method is as follows.

1. Query generation: A client sends a query with the additional information, such as the storage capacity of client and the deadline of the query, to the server through the uplink. In addition, the client begins to store the tables required for the query processing from the broadcast database.

2. Selection of the query processing method: The server calculates the response times respectively for the on-demand method, the client method, and the collaborative method, and then chooses the optimal method with the least response time.

3. Query processing: The server takes appropriate actions according to the selected processing method.

4. Data transmission to the client: The server adds the corresponding broadcast data for the client to the broadcast queue of the sub channel.

5. Data reception and reconstruction of the query result: According to the transmission data of the sub channel, the client takes the corresponding process. When receiving the query result, the client uses the result as it is; when receiving the processing rule, the client receives the tuples attached with the identifier and reconstructs the query result based on the rule; when nothing received, the client performs query processing by itself, only after receiving the necessary tuples required for the query processing via the main channel.

In the LRT method, the transmitting data of the collaborative method is added to the broadcast queue of the sub channel at the position before the transmitting data of the first on-demand method in the queue, since the size of the processing rule is much smaller than that of the data for the on-demand method. However, the server cannot choose the collaborative method, if a query processed by the on-demand method already in the queue exceeds the deadline by adding the rule for the latest query.

3) Problems of LRT Method: In the LRT method, when the selection number of the on-demand method with large query results increases, the queue may become very long.

Moreover, the response times of queries processed by the on-demand method in the queue becomes slightly longer, since the processing rules are inserted before the transmitting data of the on-demand method. When the selection number of the collaborative method increases, it is impossible to disregard the increase in the response time. Specifically, since we do not allow the queries processed by the on-demand method already in the queue exceed the deadline, the server cannot choose the collaborative method.

B. Extended Methods

In the LRT method, a query will fail when the response time exceeds the deadline even if the server chooses any method among the three methods. Therefore, we propose the following three extended methods in order to reduce the response time and heighten the success rate of the query processing.

The extended methods are based on the LRT method. The server firstly chooses the query processing method among the on-demand method, the client method, and the collaborative method. When there is no available method to be chosen, or when the response time exceeds the deadline, the server performs the specific process. Figure 3 shows the workflow of query processing in the extended methods.

1) LRT-I Method: In the LRT method, the server cannot choose the collaborative method if a query processed by the on-demand method and whose result is already in the queue

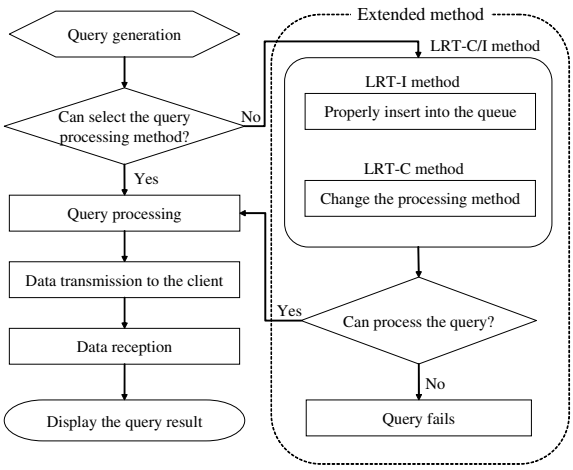

Fig. 3. Flow of query processing in the extended methods.

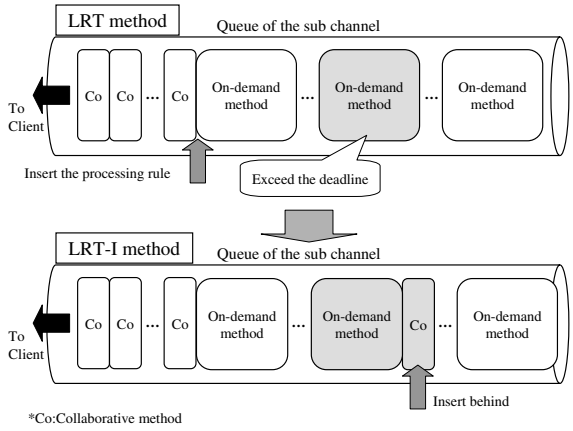

Fig. 4. LRT-I method.

exceeds the deadline by adding the rule for a query processed by the collaborative method at the front of the queue.

In the LRT-I (LRT-Inserting) method, as shown in Figure 4, the rule of the collaborative method is inserted behind the query result of the on-demand method, of which the response time exceeds the deadline in the LRT method. If there are multiple such queries, it will be inserted behind the last one. The query will fail, if its own response time exceeds the deadline by performing this procedure.

2) LRT-C Method: In the LRT method, if the selection number of the on-demand method with large query results increases, the queue will become very long.

As shown in Figure 5, the LRT-C (LRT-Change) method, the server changes the query processing method of a query from the on-demand method to the collaborative method, of which the time from receiving the processing rule to receiving the required tuples is the shortest.

The server does not perform the change when the query fails by this change.

3) LRT-C/I Method: The performance is improved further by combining the LRT-I method and the LRT-C method, since

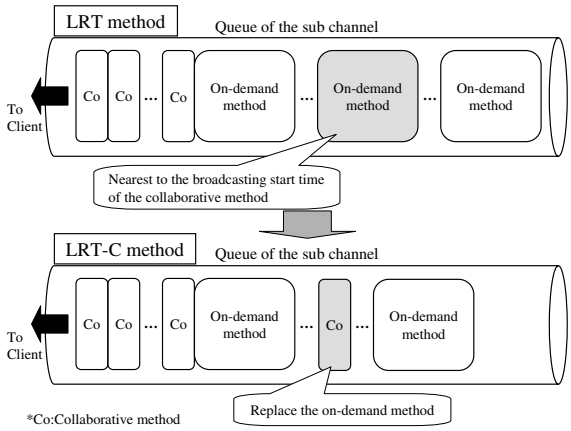

Fig. 5. LRT-C method.

these enhanced methods are independent.

The LRT-C/I (LRT-I & LRT-C) method is the combination of the LRT-I and the LRT-C methods. In the LRT-C/I method, the server totals the change in response time for all queries in the queue by applying the LRT-I and the LRT-C methods, and chooses optimum one.

IV. Evaluation

This section evaluates the performance of our methods, the LRT, LRT-I, LRT-C, and LRT-C/I method. Two evaluation criteria are introduced as follows.

Success rate: The ratio of the queries, of which clients could get the result, to all of the queries clients issued.

Response time: The elapsed time from the query generation to the acquirement of the query result. Note that the response time does not include the time for transmitting the query from the client to the server and the time for processing the data at the client side or the server side, since they are adequately short.

A. Simulation Environment

In the evaluation, the database schema and the query model is supposed for an information service in s shopping center described in Section II-A.

The database consists of a shop table and a goods table. For the sake of simplicity, all tuples are supposed to be the same size. Moreover, a client only issues queries of a natural join with the shop table and the goods table.

Table I shows the parameters and the values used in the evaluation. In the evaluation, when clients intend to store data more than the permitted size in processing query, the query fails. In addition, the deadline is given as a parameter to each query, as shown in Table II, and the query fails when the client cannot get the query result by the time of deadline. The query generation intervals are given by the exponential distribution with a parameter of query frequency.

TABLE I

PARAMETERS.

Name	Value
Size of database[KByte]	15000
Size of processing rule[KByte]	1
Bandwidth of main channel[Kbps]	4000
Bandwidth of sub channel[Kbps]	500
Number of identifier	50
Ratio of clients with shortage storage	0.5

TABLE II

DEADLINES.

Seconds	20	30	40	50	60	70
Ratio	5%	5%	10%	20%	20%	10%
Seconds	80	90	100	110	120	
Ratio	10%	5%	5%	5%	5%	

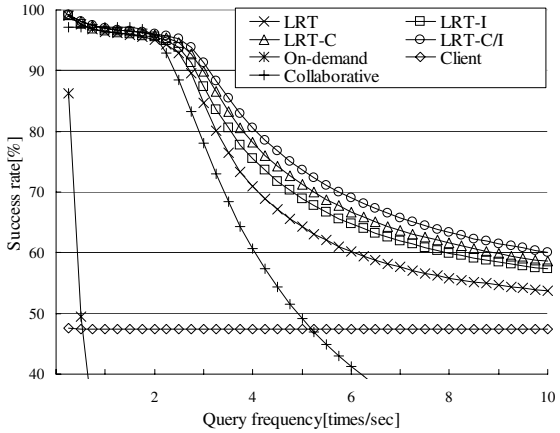

Fig. 6. Query frequency vs. success rate.

B. Simulation Result

1) Success Rate: Figure 6 shows the success rates of the query processing respectively for the LRT method, the LRT-I method, the LRT-C method, and the LRT-C/I method, compared with the on-demand method, the client method, and the collaborative method in changing the query frequency.

In the case of the on-demand method, the success rate is high when the query frequency is low; while the success rate falls suddenly when the query frequency becomes higher. This is because the sub channel is exhausted if the query frequency is so high with the large query results transmitted via the narrow band sub channel.

In the case of the client method, even if the query frequency changes, the success rate does not change since once the client stores all the necessary tables, the query processing is performed by itself. However, the success rate is quit low due to the limit of the client storage.

The success rate of the collaborative method is high when the query frequency is low; while it falls suddenly when the query frequency exceeds 2. This is on account for the lack of identifiers when the query frequency becomes higher.

The success rate of the LRT method is much higher than the on-demand method, the client method, and the collaborative method, since the server can choose the optimal method among these three methods according to the situation.

The success rate of the LRT-I method becomes rather higher than the LRT method. This is because the query which fails in the LRT method can be processed in the LRT-I method. Similarly, the LRT-C method also improves the success rate compared with the LRT method. In the case of the LRT-C method, the selection number of the collaborative method increases. Since the size of the transmitting data in the collaborative method is much smaller than that of the on-demand method, it seems to moderate the exhaustion of the sub channel, which accounts for the improvement of the success rate. Moreover, since the LRT-I method only adds the query to the queue, while the LRT-C method can shorten the queue, the server can process more queries, and the success rate of the LRT-C method becomes higher than that of the LRT-I method.

In the case of the LRT-C/I method, the success rate is highest since it takes advantage of both methods. However, the degree of the improvement is not remarkable, since the selection rate of the collaborative method increases just a little compared with that of the LRT-C method, due to the limitation of the number of identifiers.

2) Response Time: Figure 7 shows the evaluation of the average response time for all the methods.

The response time of the on-demand method is short when the query frequency is low; it becomes much longer when the query frequency increases. This is because the queue of the sub channel becomes much longer.

In the case of the client method, even if the query frequency changes, the response time does not change. Because the time to store the necessary tables for the query processing at the client does not change with the change in the query frequency.

In the case of the collaborative method, even if the query frequency changes, the response time also does not change. In the collaborative method, the server broadcasts processing rules via the sub channel and the size of a processing rule is very small, the queue does not become as long as the on-demand method. Moreover, the response time of the collaborative method is shorter than that of the client method, since the client only receives the necessary tuples.

In the case of the LRT method, the response time is shorter than the on-demand method, the client method, and the collaborative method when the query frequency is very low. However, the response time becomes longer when the query frequency increases, which is almost the same as the client method.

The response time of the LRT-I method is slightly longer than that of the LRT method. This is because the response time of the collaborative method becomes longer by inserting the processing rule into the middle of the queue, instead of inserting it in front of the queue in the LRT method.

The response time of the LRT-C method is shorter than that of the LRT method. When the server changes the query

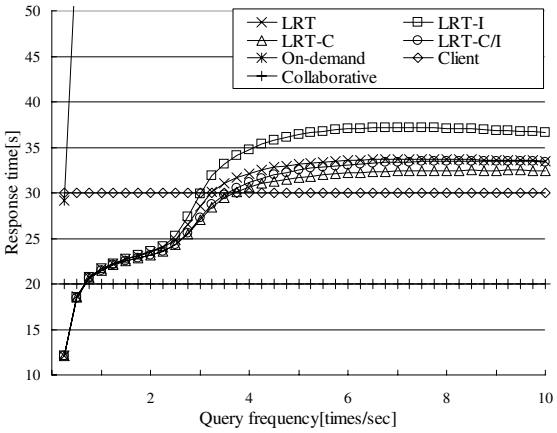

Fig. 7. Query frequency vs. response time.

processing method in the queue from the on-demand method to the collaborative method, the selection number of the collaborative method increases, and the exhaustion of the sub channel is mitigated. As a result, the average response time becomes shorter, while the response time of the changed query may become longer.

The response time of the LRT-C/I method is longer than that of the LRT-C method, but shorter than that of the LRT-I method, since it is the combination of the LRT-I method and the LRT-C method. As the query frequency becomes higher, the response time approaches to the mean value of the LRT-I method and the LRT-C method.

3) Discussions: The simuration results show that the LRT-C/I method should be used when many users attach more importance to the success rate, while the LRT-C method should be used when many users attach more importance to the response time.

V. Related Works

Many researches have been done to improve the performance of the broadcast database system. Broadcast disks [1] reduces the response time by frequently broadcasting hot items. The idea of the broadcast disks can be introduced into our research, for example, the server frequently broadcasts the table with high access frequency if the table of the database is considered as an item. [3] is similarly to our research, which unites the push-based broadcast and pull-based broadcast, the broadcast channel being divided into the push and pull channel. They use the pull channel to broadcast the required data item, while our method is to broadcast the query result and the processing rule.

VI. Conclusions

In this paper we proposed a new query processing method which dynamically chooses the query processing method for broadcast database system. In the proposed method, the server chooses the method with the least response time among the three query processing methods. Furthermore, our extended methods pay more attention to the broadcast queue at the server, and improve the efficiency of the query processing by changing the order of the submitted queries in the queue and changing the processing methods for the queries. Additionally, the simulation results confirmed that the proposed methods achieved the high performance in the success rate of query and the average response time compared with the traditional methods.

In the future, we plan to examine a method to choose the query processing method dynamically according to the query generation frequency. Furthermore, we also plan to examine a method to choose optimal query processing method under the situation where the query interval changes dynamically.

Acknowledgments

This research was partially supported by The 21st Century Center of Excellence Program “New Information Technologies for Building a Networked Symbiotic Environment” and Grant-in-Aid for Young Scientists (A)(16680005) and for Scientific Research (A)(17200006) and (B)(2)(15300033) of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] S. Acharya, M. Franklin, and S. Zdonik, “Broadcast disks: Data management for asymmetric communication environments,” Proc. ACM SIGMOD’95, pp.199–210, May 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik, “Disseminating updates on broadcast disks,” Proc. VLDB’96, pp.354–365, Sept. 1996.

[3] S. Acharya, M. Franklin, and S. Zdonik, “Balancing push and pull for data broadcast,” Proc. ACM SIGMOD’97, pp.183–194, May 1997.

[4] D. Aksoy, M. Franklin, and S. Zdonik, “Data staging for on-demand broadcast,” Proc. VLDB’01, pp.571–580, Sept. 2001.

[5] Q. Hu, D. Lee, and W. Lee, “Performance evaluation of a wireless hierarchical data dissemination system,” Proc. Mobicom’99, pp.163–173, Aug. 1999.

[6] M. Kahita, T. Terada, T. Hara, M. Tsukamoto, S. Nishio, “A collaborative query processing method for a database broadcasting system,” Proc. IASTED Int’l Conf. on Communications, Internet and Information Technology (CIIT’02), Nov. 2002.

[7] E. Yajima, T. Hara, M. Tsukamoto, and S. Nishio, “Scheduling strategies of correlated data in push-based systems,” Information Systems and Operational Research (INFOR), Vol.39, No.2, pp.152–173, May 2001.

[8] E. Yajima, T. Hara, M. Tsukamoto, and S. Nishio, “Scheduling and caching strategies for broadcasting correlated data,” Proc. ACM Symposium on Applied Computing (ACM SAC’01), pp.504–510, March 2001.