A Query Processing Method Considering the Characteristic of Energy Consumption for Broadcast Database Systems

Shinya KITAJIMA† Takahiro HARA† Tsutomu TERADA‡ Tomoki YOSHIHISA† Shojiro NISHIO†
†Dept. of Multimedia Eng., Grad. School of Information Science and Technology, Osaka University
Email: {kitajima.shinya, hara, yoshihisa, nishio}@ist.osaka-u.ac.jp
‡Dept. of Electrical and Electronics Eng., Grad. School of Science and Technology, Kobe University
Email: tsutomu@eedept.kobe-u.ac.jp

Abstract

In recent years, there has been an increasing interest in a broadcast database system where the server periodically broadcasts contents of a database to mobile clients such as PDAs and smartphones. There are three query processing methods in the broadcast database system. Generally, mobile clients have limits in energy consumption, i.e., battery, and each of the three methods consumes different amount of energy for query processing. In this paper, we propose a new method which dynamically chooses one of the three query processing methods considering energy consumption.

1 Introduction

The recent evolution of wireless communication technologies has led to an increasing interest in broadcast information systems in which data are disseminated via the broadcast. In such systems, a server broadcasts various data periodically via the broadband channels, while clients pick out and store necessary data. There are many studies for improving the performance of broadcast information systems[1, 2, 3]. 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.

In this paper, we assume a broadcast system that the server periodically broadcasts contents in a database and clients issue queries to retrieve data from the database. We call such a system *broadcast database system*. There are three basic query processing methods in this system, on-demand, client, and collaborative methods. However, the performance of each method changes according to the system situation such as query frequency.

In [5], we proposed a query processing method which chooses the method with the least response time among the three basic methods. In this paper, based on the method proposed in [5], we propose a new method which considers the energy consumption of mobile clients when choosing a query processing method. Furthermore, our simulation evaluation reveals that the proposed method improves the lifetimes of clients with a low remaining battery power.

The remainder of this paper is organized as follows. Section 2 describes the outline of a broadcast database system and introduces three basic query processing methods and the conventional method in the broadcast database system. Section 3 explains our method in details. Section 4 evaluates the performance of our method. Finally, we conclude the paper in Section 5.

2 Broadcast Database System

Figure 1 illustrates the concept of a broadcast database system. In this system, the server broadcasts contents in a relational database via the broadcast channel and processes queries from clients. Clients issue queries to retrieve the necessary data from the database. Clients have a small storage, low power resource, and low CPU capability, such as a PDA. The broadcast channel from the server to clients is divided into two channels: a broadband *main channel* to disseminate the contents in a database repeatedly, and a narrowband *sub channel* to disseminate the other data. Moreover, there is a narrowband uplink channel from clients to the server. Clients use the uplink channel to send queries to the server.

2.1 Assumed Environment

We assume that our method is used for disseminating information to many and unspecified users. For example, for an information service in a shopping center, the server broadcasts data in the database including advertising information, shop information, and goods information in the shopping center, while thousands of users receive the broadcast information and retrieve the necessary information. The broadcast data include maps of shops and images of goods. The bandwidth to broadcast is about 10 Mbps.

978-0-7695-3575-3/09 $25.00 © 2009 IEEE
DOI 10.1109/CISIS.2009.16

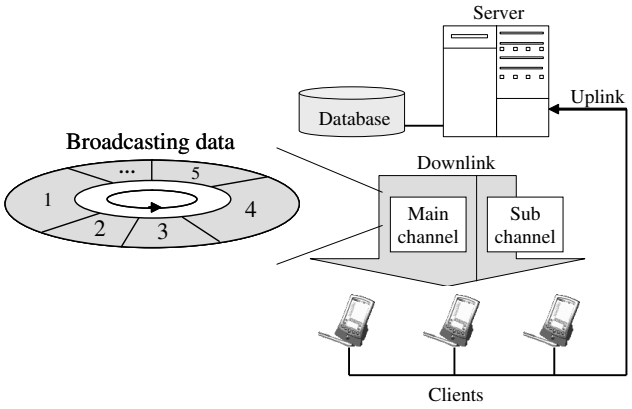

Figure 1. Broadcast database system.

The server always broadcast the contents of the database and the index of the broadcast schedule periodically.

The users sometimes 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 to the shop selling the item". We assume several minutes' delay for receiving query result is acceptable for clients. On the other hand, users set the deadline of the response time to each query. When users cannot receive query result by the time of deadline, the query fails. Although users stay in the shopping center for a certain time, the battery is limited and the users who run out of the battery cannot receive the service.

2.2 Query Processing Methods

In the broadcast database system, there are three basic query processing methods and one adaptive method as follows.

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, the query is processed by the server completely, and not much workspace is required for query processing at client. Thus the energy consumption is low.

Client method: A client stores all the tables related to the query, and processes the query by itself. Query processing causes a heavy workload on the client. This consumes a lot of battery power.

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. Using the received rules, the client receives the necessary tuples via the main channel referring to the identifiers, and reconstructs the query result by combining these tuples[4].

LRT (Least Response Time) method: The system performance, when each method is used individually, changes with the environmental conditions such as query frequency. In the LRT method[5], when the server receives a query, 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.

3 ELEC Method

In the previously proposed LRT method, there is a problem that it might shorten lifetimes of clients with a remaining low battery, since it aims to improve the success rate and the response time of the query, but does not consider the energy consumption. Therefore, we propose a new query processing method, called ELEC (Extended LRT considering Energy Consumption) method which considers remaining batteries of mobile clients when choosing one of the three baic query processing methods. This method improves the lifetimes of clients with low remaining battery.

3.1 Outline

In the ELEC method, the server basically chooses a query processing method according to the LRT method. However, the server checks the remaining battery of the client issuing the query, and preferentially chooses the method with the lowest energy consumption when the remaining battery is lower than the threshold P_{TH}. The procedure of the query processing is as follows.

1. If the remaining battery of the client which currently issues a query is more than P_{TH}, the server chooses the query processing method according to the LRT method.

2. Otherwise, the server chooses the possible query processing method with the lowest energy consumption.

The optimal P_{TH} will change according to the query frequency, remaining battery of clients, and so on.

3.2 Calculation of the Thresholds

For deciding the appropriate threshold P_{TH}, the server repeatedly examines last q queries with changing a tentative value of the threshold. Then, the server decides the threshold, so that the energy consumption for clients whose remaining battery is less than the threshold becomes least. The detailed procedure is as follows.

1. We denote the remaining battery of the client which issues the last x-th query Q_x as P_x. We also denote the sequence of numbers which consists of q elements as $A = (P_1, P_2, \ldots, P_q)$, and the ascending sequence of A as the candidate threshold sequence A'.

2. We represent the k-th element of A' as $A'[k]$. The server takes the following steps for $A'[i](i = 1, 2, \ldots, q)$.

(2a) The server examines last q queries by setting $A'[i]$ as a tentative threshold.

(2b) The server calculates the energy consumption per query of clients whose remaining battery is less than $A'[i]$, which is represented as E_i.

(2c) If $E_j > E_{j-1} > E_{j-2} (j = 1, 2, \ldots, q)$, the server judges the appropriate threshold is less than A_j, and goes to step 3.

3. We represent the minimum of $E_i (i = 1, 2, \ldots, q)$ as E_{min_i}, and decides the threshold for next q queries as $A'[min\_i]$.

To examine last q queries in step (2a), the server maintains the information of last q queries, the state of the queue of the sub channel when the last q-th query is issued, and the information of identifiers being used in the collaborative method.

3.3 Actual Energy Consumption of Mobile Terminal

To determine the query processing method with the lowest energy consumption, we need to calculate the energy consumption of each method for a given query. For this aim, we investigate the energy consumption of various operations by using an actual mobile terminal (iPAQ rx5965, Hewlett-Packard Development Co.). To measure the energy consumption, we use the remaining battery shown by the operation system. It displays the ratio of the remaining battery against the full battery. We can know the energy consumption of a specific operation such as reading and writing by measuring the remaining battery after several time.

We implemented a tool on the terminal which periodically records the remaining battery at a fixed time interval while performing a specific operation repeatedly, e.g., receiving data using wireless LAN or writing data to the ROM. Moreover, we implemented a broadcast server and a receiving application to investigate the energy consumption for receiving broadcast data. The broadcast contents are supposed for an information service in a shopping center as described in Section 2.1.

The database schema has a shop table {shopID, shop name, image, ...} and a goods table {goodsID, shopID, goods name, image, ...}. Furthermore, shops are classified into several genres according to selling goods. The server first broadcasts a shop table, then broadcasts a goods table for each genre. We denote the time required to broadcast tables of all genres as the *broadcast cycle*.

Table 1 shows the elemental energy consumption ratio derived from the experimental result. Here, the elemental energy consumption ratio is defined as the energy consumption for performing a particular operation divided by the operating time. In Table 1, we also define the symbols used in this paper. Moreover, we show the value which is obtained by dividing each elemental energy consumption ratio by that of the idle case (we call it "ratio to idle").

Table 1. Elemental energy consumption ratio.

		Value	Ratio to idle
Idle	E_I	13.2	1.0
Wireless LAN	E_L	47.2	3.6
Reading	E_R	23.0	1.7
Writing	E_W	21.0	1.6
Downloading	E_D	68.4	5.2
Receiving broadcast data	E_B	71.6	5.4
Combining tuples	E_C	23.6	1.8
Wireless LAN & Writing	E_{LW}	69.6	5.3
Wireless LAN & Uniting	E_{LU}	71.6	5.4

3.4 Formulation of Energy Consumption

In this sub section, we formulate the energy consumption for operations and query processing using the ratio to idle shown in Table 1. The energy necessary to process a query represents the energy required to receive necessary tables and to combine the tuples after issuing the query.

3.4.1 Elemental energy consumption

First, we define the energy consumption for CPU, wireless LAN, reading, and writing. Here, t denotes the time during which the client is engaged in the corresponding operation.

Elemental energy consumption: We call the energy consumption when the terminal is switched on but does nothing, i.e., idle as the elemental energy consumption (E_I), which is define as follows:

$$E_I(t) = t. \tag{1}$$

Wireless LAN: The energy consumption for Wireless LAN (E_L) is defined as follows:

$$E_L(t) = 3.6\,t. \tag{2}$$

Reading and writing: The energy consumptions for reading (E_R) and writing (E_W) are defined as follows:

$$E_R(t) = 1.7\,t, \tag{3}$$

$$E_W(t) = 1.6\,t. \tag{4}$$

Combining tuples: The energy consumption for combining tuples (E_C) is defined as follows:

$$E_C(t) = 1.8\,t. \tag{5}$$

3.4.2 Energy consumption for basic operations

Using the elemental energy consumption in Section 3.4.1, we define the energy necessary to receive broadcast data, to process data, and to read and to write data.

Receiving broadcast data: We define the energy consumption for client to receive broadcast data and writes the data to its disk as follows:

$$E_{rcv}(t) = E_I(t) + E_L(t) + E_W(t) = 6.2\,t. \quad (6)$$

Combining tuples: We define the energy consumption to combine tuples as follows:

$$E_{tpr}(t) = E_I(t) + E_C(t) = 2.8\,t. \quad (7)$$

Writing data: The energy necessary to write data to disk (without receiving broadcast data) is defined as follows:

$$E_{wrt}(t) = E_I(t) + E_W(t) = 2.6\,t. \quad (8)$$

Reading data: The energy necessary to read data before processing is defined as follows:

$$E_{rd}(t) = E_I(t) + E_R(t) = 2.7\,t. \quad (9)$$

3.4.3 Energy consumption for query processing

In this clause, we define the energy necessary to process a query with the on-demand method, the client method, and the collaborative method, respectively.

On-demand method: In the on-demand method, client only receives the query result from the sub channel. Therefore, the energy necessary to process a query with the on-demand method (E_{on}) is represented as follows:

$$E_{on}(t) = E_{rcv}(t) = 6.2\,t. \quad (10)$$

Client method: In the client method, after receiving the tables necessary to process the query from the main channel, the client processes the query. Therefore, the energy necessary to process the query with the client method (E_{cl}) is represented as follows, by using the time to receive the necessary tables t_{cl}^{rcv}, the time to read these tables t_{cl}^{rd}, the time to combine the tuples t_{cl}^{prc}, and the time to write the query result to the disk t_{cl}^{wrt}:

$$\begin{aligned} E_{cl}(t) &= E_{rcv}(t_{cl}^{rcv}) + E_{rd}(t_{cl}^{rd}) \\ &\quad + E_{tpr}(t_{cl}^{prc}) + E_{wrt}(t_{cl}^{wrt}). \end{aligned} \quad (11)$$

Collaborative method: In the collaborative method, after the client receives the processing rule, it receives only the tuples attached with the identifiers and combines the tuples based on the processing rule. Therefore, the energy necessary to process the query with the collaborative method (E_{co}) is represented as follows, by using the time to receive the processing rule $t_{co}^{rcv_{rule}}$, the time to read the processing rule $t_{co}^{rd_{rule}}$, the time to receive the tuples necessary to process the query $t_{co}^{rcv_{tpl}}$, the time to read these tuples $t_{co}^{rd_{tpl}}$, the time to combine the tuples t_{co}^{prc}, and the time to write the query result to the disk t_{co}^{wrt}:

$$\begin{aligned} E_{co} &= E_{rcv}(t_{co}^{rcv_{rule}}) + E_{rd}(t_{co}^{rd_{rule}}) + E_{rcv}(t_{co}^{rcv_{tpl}}) \\ &\quad + E_{rd}(t_{co}^{rd_{tpl}}) + E_{tpr}(t_{co}^{prc}) + E_{wrt}(t_{co}^{wrt}). \end{aligned} \quad (12)$$

While these formula are intuitive and simple, these are enough to estimate the energy consumption in our proposed method.

4 Evaluation

This section evaluates our proposed ELEC method. The following three evaluation criteria are used for the evaluation.

Lifetime: The average of the elapsed times from user's arrival to the user's exit, due to either spending the predetermined time or running out of the battery. Note that we assume that clients consume battery only by operations related to query processing.

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

Response time: The average elapsed time from a query generation to the receipt of the query result for successful queries. Note that the response time includes neither the time for transmitting a query from a client to the server nor the time for processing the data at the client side or the server side, since they are short enough.

4.1 Simulation Environment

In the evaluation, the database schema and the query model is as the same as described in Section 3.3.

In the evaluation, the deadline is given as a parameter to each query, and the query fails when the client cannot get the query result until the deadline. Furthermore, the ratio of clients who cannot store all the necessary tables for processing queries is given. If clients are required to store the data more than the capacity, the query also fails.

Table 2 shows the parameters used in the evaluation. The rate of necessary tuples represents the rate of tuples which are included in the query result to all tuples in the table. The user arrival intervals are given by the exponential distribution with a parameter on the user arrival frequency. Each user has a mobile client to receive the service. The initial

Table 2. Parameters.

Parameter	Value
Simulation time[sec]	36000
Estimated staying time of users[sec]	7200
User arrival frequency[/sec]	20
Query interval for a user[sec]	300
Deadline of response time[sec]	80
Number of genres	10
Number of shops for a genre	5-10
Number of goods for a shop	200
Size of a tuple[KByte]	10
Number of identifiers	200
Bandwidth of main channel[Mbps]	10
Bandwidth of sub channel[Mbps]	2
Size of a processing rule[KByte]	1
Average ratio of necessary tuples	0.03
Standard deviation of necessary tuples	0.01
The initial remaining battery[unit energy]	200 - 2000
The storage capacity[MB]	1 - 100
The frequency of CPU[MHz]	200 - 500
The speed of writing data[MB/s]	0.6
The speed of reading data[MB/s]	3.5

Table 3. Success rate and response time.

Method	Success rate	Average response time
Random	85.27%	62.86[sec]
Minimum Energy	85.84%	63.85[sec]
LRT	99.80%	45.76[sec]
ELEC	99.09%	49.43[sec]

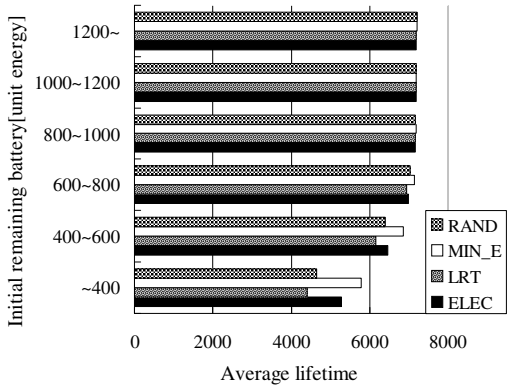

Figure 2. User arrival frequency and average lifetime.

remaining battery of each client is given by the uniform distribution from 200 to 2000 unit energies, the storage capacity of each client is given by the uniform distribution from 1 to 100 MB, and the spec of CPU is given by the uniform distribution from 200 to 500 MHz. A user, who arrives at the shopping center, issues a query according to the query interval with the deadline of the response time, the storage capacity, and the remaining battery. Each user exits the shopping center after spending the estimated staying time.

We define t_{cl}^{prc} in formula (11) is defined as follows since t_{cl}^{prc} on iPAQ rx5965 (CPU : 400MHz) is measured as 1.8[sec]:

$$t_{cl}^{prc} = 1.8 \times \frac{400}{CPU[MHz]}. \quad (13)$$

Moreover, t_{co}^{prc} in formula (12) is defined as follows,

$$t_{co}^{prc} = t_{cl}^{prc} \times r. \quad (14)$$

4.2 Methods for Comparison

In our simulations, we compared our proposed method with the following two methods and the LRT method.

Random Method: In the random method, the server chooses the query processing method randomly.

Minimum Energy Method: In the minimum energy method, the server chooses the query processing method with the lowest energy consumption. This method is the same as the ELEC method whose P_{TH} is set to the maximum value.

4.3 Simulation Results

4.3.1 Comparison with Other Methods

Table 3 shows the evaluation results of the success rate and the average response time of the random method, the minimum energy method, the LRT method and the ELEC method. Moreover, Figure 2 shows the evaluation results of the average lifetime under different initial remaining battery conditions of clients, which is classified by every 200 unit energies. The parameter q of the ELEC method is set to 50 according to the preliminary experiment.

Table 3 and Figure 2 show that the average lifetime of the minimum energy method is the longest among the four methods, though the success rate and the average response time is much worse than those of the LRT method and the ELEC method. In the minimum energy method, since the server considers only the enegy consumption of the client, the average lifetime becomes long. However, the restriction of the sub channel, storage, and identifier worsens the success rate and the average response time, due to continuously choosing the query processing method with the lowest energy consumption (the on-demand method in most case).

The average lifetime of the ELEC method is much longer than that of the LRT method, and the degrades in the success rate and the response time are little. In the ELEC method,

Table 4. Impact of q on success rate and response time.

q	Success rate	Average response time	Average threshold
10	95.47%	55.67[sec]	544[unit energy]
50	99.09%	49.43[sec]	133[unit energy]
100	99.43%	48.28[sec]	76[unit energy]

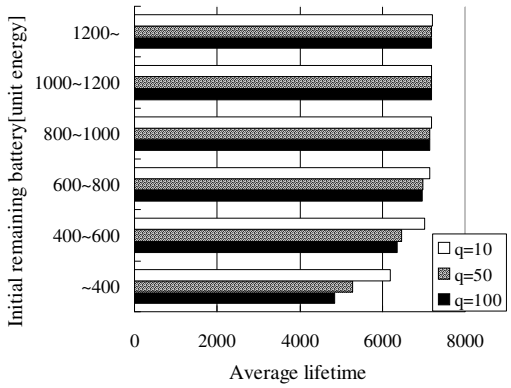

Figure 3. Impact of q on average lifetime.

the lifetime of clients which have a low remaining battery becomes long, since the server preferentially chooses the method with the lowest energy consumption when the remaining battery of the client is lower than the threshold.

4.3.2 Impact of q

Table 4 shows the success rate and the average response time of the ELEC method when q is 10, 50, and 100. Figure 3 shows the evaluation results of the average lifetime under different initial remaining battery conditions of a client, which is classified by every 200 unit energies.

In the case where $q = 10$, the average lifetime of clients with a low remaining battery is longer than that in the case where $q = 50$, while the success rate and the average response time are worse. When q is small, the number of the candidate thresholds (A') is also small, and the threshold (P_{TH}) tends to become high. Therefore, the server often chooses the query processing method with the least energy consumption, the average lifetime of clients with a low remaining battery becomes longer. On the other hand, in the case where $q = 100$, the success rate and the average response time are almost the same as the case where $q = 50$, while the average lifetime of clients with a low remaining battery is shorter. When q is large, the number of the candidate thresholds is also large, and the threshold tends to become low. Therefore, the server can rarely choose the query processing method with the lowest energy consumption. As a result, the average lifetime of clients with a low remaining battery becomes short.

5 Conclusions

In this paper, we proposed a new query processing method which dynamically chooses a query processing method by considering the energy consumption. In the proposed method, the server preferentially chooses the method with the lowest energy consumption when the remaining battery is lower than the threshold. The simulation results revealed that the proposed method improved the lifetime of clients with a low remaining battery while keeping high success rate and short average response time compared with the LRT method.

In future, we plan to extend our proposed method to deal with the similarity among queries and packet losses.

Acknowledgments. This research was partially supported by "Global COE (Centers of Excellence) Program", Special Coordination Funds for Promoting Science and Technology "Formation of Innovation Center for Fusion of Advanced Technologies: Yuragi Project", and Grant-in-Aid for Scientific Research (18049050) of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] Acharya, S., Alonso, R., Franklin, M., and Zdonik, S.: Broadcast disks: data management for asymmetric communication environments. Proc. ACM SIGMOD'95, pp. 199–210, May 1995.

[2] Acharya, S., Franklin, M., and Zdonik, S.: Balancing push and pull for data broadcast. Proc. ACM SIGMOD'97, pp. 183–194, May 1997.

[3] Aksoy, D., Franklin, M., and Zdonik, S.: Data staging for on-demand broadcast. Proc. VLDB'01, pp. 571–580, Sept. 2001.

[4] Kashita, M., Terada, T., Hara, T., Tsukamoto, M., and Nishio, S.: A collaborative query processing method for a database broadcasting system. Proc. CIIT'02, pp. 60–66, Nov. 2002.

[5] Kitajima, S., Cai, J., Terada, T., Hara, T., and Nishio, S.: A query processing mechanism based on the broadcast queue for broadcast database systems. Proc. ISWPC'06, pp. 450–455, Jan. 2006.