Scheduling for ordered query services in multi-channel data broadcast systems

Yu-Hsien Lin, Yi-Syuan Jiang, and Wei-Mei Chen
Department of Electronic Engineering
National Taiwan University of Science and Technology
Taipei 106, Taiwan
{M9602109, B9630132, wmchen}@mail.ntust.edu.tw

***Abstract*—In wireless mobile environments, the bandwidth and client capability are limited. Data broadcasting is an efficient and scalable method to disseminate information to a large amount of mobile clients. In recent years, several research has engaged in scheduling dependent data on multiple broadcast channels. However, most of the previous research focused on unordered queries and only few of them on ordered queries. In this paper, we study the problem of data allocation for ordered queries on multiple broadcast channels and devise an efficient algorithm based on a local search strategy. The experimental results show that the proposed algorithm outperforms the existing algorithms when the query distribution is known. Moreover, our algorithm with minor modification can also be applied efficiently when the query distribution is unknown.**

Keywords: data broadcasting, data scheduling, multiple channels, wireless mobile networks

I. Introduction

Nowadays, due to the advances in wireless networks, people can access information by portable devices such as laptops, PDA, and smart phones in daily life. In this scenario, there are various media contents provided, including stock information, weather forecast, traffic report, and news [7][21]. Because the bandwidth and client capability are restricted[15], disseminating information efficiently under limited resources conditions is an important issue in wireless mobile environments.

Data broadcasting is an efficient and scalable method to disseminate information for a large amount of mobile clients. Typically, there are three kinds of data broadcast models: the pull-based model, the push-based model, and the hybrid broadcast model [13][15]. In the pull-based model (also refer to on-demand broadcast), clients send requests to the server and keep listening broadcast channels for the data requested. In the push-based broadcast, servers broadcast all data periodically and clients retrieve the data needed when it is broadcasted. Based on this model, the scheme of broadcast disks [1][2] schedules data on single channel by the access probability in which hot data items are broadcast more frequently. The hybrid broadcast model applies pull-based broadcast and push-based broadcast in different situations. For instance, the hybrid approach suggested in [26] broadcasts popular data by the push-based method and others by the pull-based model.

In most applications, clients request several kinds of data in a query, which is called complex query (or dependent data). In general, complex queries are categorized into two types: unordered queries and ordered queries. Data in unordered queries does not have to be retrieved in certain order. The QEM algorithm [5] schedules unordered queries according to the access probability. This problem is proved to be NP-hard in [6]. Later, the MQEM algorithm is suggested in [13] which improves the performance of the QEM algorithm by using a weighted undirected graph. Another scheduling algorithm is proposed by using tree structures to allocate data on multiple channels [23]. On the other hand, a genetic algorithm in [8] is proposed to solve the data scheduling problem for unordered queries on multiple channels.

In ordered query environments, data items are associated such that objects need to be retrieved in sequential order. In the systems with only one single channel, there are some researches on allocating ordered queries by various directed graphs and data structures in [4][14][25]. For multiple channel environments, Huang and Chen [10] use a genetic algorithm to allocate ordered data on multiple channels. The algorithms, MPPA, RMPPA, and CMPPA [17][18] arrange data items on multiple channels by exploiting DAG and heaps. Given the query distribution, Hung and Chen present the MULS framework [11] to schedule sequential data broadcasting. The MULS algorithm contains two stages: OLS and BASIC. The OLS algorithm checks backward or forward for two consecutive items item to allocate better placement. The BASIC algorithm is proposed to schedule the broadcast program by comparing all possible pairs of data and then exchanging them for the best arrangement. Although MULS can adjust the broadcast program for less access time, it takes much time to evaluate possible solutions. In addition, the broadcast scheduling problem with data replication is studied for ordered queries in [9].

Most researches consider the environment is static and the exact request distribution is known. However clients may sent queries in any time or new clients might join in the system spontaneously. Under such dynamic conditions, the related query distribution is hard to obtain in advance. In this paper, we allocate ordered queries on multiple channels when the query distribution is unknown. Our algorithm schedules the ordered data item in a query and adjusts the scheduled data item when they appear in the unscheduled query again. We first exam and evaluate some promising data items, and then exchange the broadcast sequence to reduce the average access

time. Meanwhile, when the query distribution is known, our algorithm takes less average access time and reduces execution time, compared to the MULS algorithm.

The rest of this paper is organized as follows. Section 2 describes the environment, preliminaries, and analytical models. In Section 3, the proposed algorithm is present and brief examples are given. The simulation results are shown in Section 4. Finally, this paper concludes with Section 5.

II. Definition

In this section, we introduce the problem definition and some notations used in this paper.

A. Environment

The environment of this paper is shown in Fig. 1. The clients send their queries to the server. Every query contains several data items which are equal in size and are ordered. When the server receives queries, it starts to schedule the broadcast program and then follows the program to broadcast data cycle by cycle. On the other hand, clients listen to all channels and download data in order when the needed data is broadcasted. If one broadcast cycle is finished and there are new data requested, the server stops broadcasting and adjusts the broadcast program. Once the broadcast program is altered, the sever informs all clients in this system and starts to broadcast by the new broadcast program.

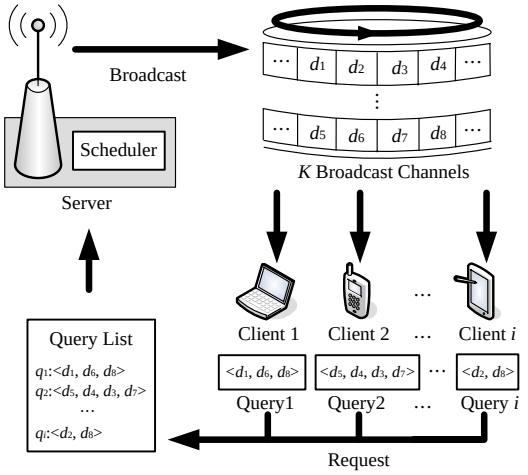

Fig. 1. The environment of broadcasting ordered queries on multiple channels.

B. Notations

Suppose that the number of channels in the system is K for $K \geq 1$ and there are a set of N distinct data items D, where $D = \{d_1, d_2, \ldots, d_N\}$ and each data item d_i is equal in size for $1 \leq i \leq N$. Let Q denote a set of all ordered queries and $Q = \{q_1, q_2, \ldots, q_M\}$ where M is the number of ordered queries requested from clients. Each ordered query contains several data items and these data items need to be downloaded in order. Let q be a query in Q, $q(j)$ be the jth data item in q, n_q be the number of data items requested in q, and f_q be the access probability of q.

Assume that the length of broadcast cycle is L, which means that for each cycle there are L data items to be broadcast in a channel. Since every data item is just broadcast once and every channel broadcasts the same number of data, we have $L = \lceil N/K \rceil$. The broadcast program is recorded as a $K \times L$ matrix P. For the data item d_i, let $\mathcal{K}(d_i)$ be the channel that d_i is arranged and $\mathcal{L}(d_i)$ be the position of d_i in channel $\mathcal{K}(d_i)$. Thus $P[\mathcal{K}(d_i)][\mathcal{L}(d_i)] = d_i$. For the reader's convenience, all of the symbols in this paper are summarized in Table I.

TABLE I
Description of parameters.

Parameters	Description
D	a set of all data items
N	the number of data items in D
d_i	the ith data items in D, $1 \leqslant i \leqslant N$
Q	a set of ordered queries
M	the number of ordered queries in Q
q	an ordered query in Q
n_q	the number of data items requested in q
f_q	the access probability of q
K	the number of broadcast channels
L	the length of a broadcast cycle
P	the broadcast program
$\mathcal{K}(d_i)$	the channel of d_i
$\mathcal{L}(d_i)$	the position of d_i in channel $\mathcal{K}(d_i)$

For example, consider that the set of all data items $D = \{d_1, d_2, d_3, d_4, d_5, d_6\}$, the number of data items $N = 6$, and $Q = \{q_1, q_2, q_3\}$. The ordered queries profile is given in Table II. If $K = 2$ and $L = 3$, three feasible broadcast programs are shown in Fig. 2.

TABLE II
Example of ordered queries profile.

Probability	Query	Contents
0.5454	q_1	$< d_1, d_2, d_3 >$
0.2727	q_2	$< d_3, d_4, d_5, d_6 >$
0.1818	q_3	$< d_3, d_4, d_6 >$

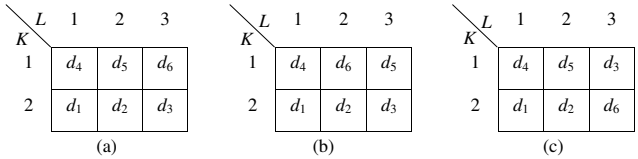

Fig. 2. Three feasible broadcast programs.

C. Analytical Models

Herein we describe some related definitions, the more detailed discussions for ordered queries on multiple channels are studied in [10][11].

When each data item is equal in size and the download bandwidth is stable, the time of download each data item is the same. Let the time of download a data item be a unit of time. In the broadcast program P for two distinct data items

d_i and d_j, the distance from d_i to d_j, denoted by $\Delta(d_i, d_j)$ is defined as follows:

$$\Delta(d_i, d_j) = \begin{cases} \mathcal{L}(d_j) - \mathcal{L}(d_i) - 1, & \text{if } \mathcal{L}(d_j) > \mathcal{L}(d_i), \\ L - \mathcal{L}(d_i) + \mathcal{L}(d_j) - 1, & \text{otherwise.} \end{cases}$$

In particular, $\Delta(d_i, d_i) = 0$.

Let $T_A(q)$ be the access time of query q; that is,

$$T_A(q) = T_S(q) + T_W(q) + T_R(q),$$

where the start-up time $T_S(q)$ is the time between the client starts listening and the client receives the first data item of query. The waiting time $T_W(q)$ is defined by $T_W(q) = \sum_{j=1}^{n_q - 1} \Delta(q(j), q(j+1))$. The retrieval time $T_R(q)$ is the time that the client spends to download all data items.

Consider the broadcast program as shown in Fig. 3. A query $q = < d_3, d_4, d_6 >$ is requested at the beginning of the ℓth cycle. The start-up time $T_S(q)$ is 2. After d_3 is downloaded, d_4 and d_6 are downloaded in the $(\ell+1)$th cycle . In this situation, the waiting time $T_W(q)$ is 0 and $T_R(q) = 3$. Then

$$T_A(q_3) = T_S(q_3) + T_W(q_3) + T_R(q_3) = 2 + 0 + 3 = 5.$$

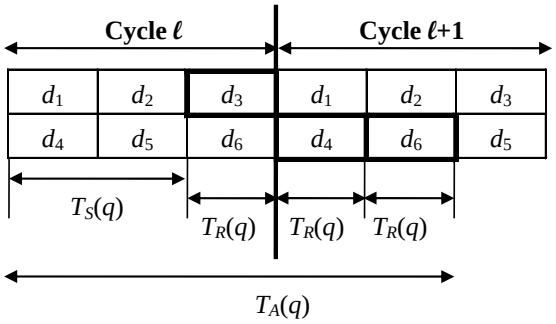

Fig. 3. A broadcast program.

Generally, clients may sent queries in any time or new clients may join in the system spontaneously. The server can not know the access probability of queries when the system is initialized. So the broadcast program needs to be updated when some new queries are sent by clients. Assume that the server updates the broadcast program if necessary and the server uses the new broadcast program for the next broadcast cycle.

III. Scheduling

In this section, we will first study the case that the query distribution is known, then extend to the case that the query distribution is unknown.

A. The case that the query distribution is known

If the query distribution is known, the server first sorts it according to the access probability. Then we have the access probability of data items as shown in Table III. Since the server knows all access probability of queries, the average access time of all query $T_A(Q)$ can be computed by

$$T_A(Q) = \sum_{i=1}^{M} f_{q_i} \cdot T_A(q_i).$$

TABLE III
A QUERY DISTRIBUTION.

Probability	Query	Contents
0.3679	q_1	$< d_7, d_{11}, d_5, d_1, d_{10} >$
0.1839	q_2	$< d_1, d_{12}, d_3, d_{10}, d_7 >$
0.1226	q_3	$< d_{12}, d_2, d_5, d_3, d_6, d_2 >$
0.0919	q_4	$< d_8, d_6, d_{12}, d_3, d_9 >$
0.0735	q_5	$< d_9, d_8, d_3, d_{10} >$
0.0613	q_6	$< d_1, d_3, d_4, d_7, d_6 >$
0.0525	q_7	$< d_2, d_6, d_{12}, d_{11} >$
0.0459	q_8	$< d_{11}, d_8, d_3, d_{10}, d_4 >$

Instead of evaluating possible combinations pair by pair, we just check a pair of consecutive data items and the relations of all scheduled queries, then adjust them one by one into some free placements for reducing the average access time for all scheduled queries. Since the channel of data item does not affect the access time, the number of free placements that we check is always less then L. Thus the computation complexity is much lower than that of MULS. The proposed algorithm is described as follows.

Algorithm 1 `schedule1`

Require: Q: a query set with $M = |Q|$
D: a data item set with $N = |D|$
K: the number of channels
A: the corresponding probability matrix
Ensure: The broadcast program matrix P

1. Sort Q according to the access probability in decreasing order.
2. $L = \lceil N/K \rceil$
3. P is a $K \times L$ matrix
4. **for** $i = 1$ to M **do**
5. **for** $j = 1$ to n_{q_i} **do**
6. **if** $q_i(j)$ is unscheduled and $q_i(j+1)$ is unscheduled **then**
7. `insertAfter`$(q_i(j), L, P)$
8. **end if**
9. **if** $q_i(j)$ is scheduled and $q_i(j+1)$ is unscheduled **then**
10. `check`$(q_i(j), P)$
11. `insertAfter`$(q_i(j+1), \mathcal{L}(q_i(j)), P)$
12. **else if** $q_i(j)$ is unscheduled and $q_i(j+1)$ is scheduled **then**
13. `check`$(q_i(j+1), P)$
14. `insertBefore`$(q_i(j), \mathcal{L}(q_i(j+1)), P)$
15. **else if** $q_i(j)$ is scheduled and $q_i(j+1)$ is scheduled **then**
16. `check`$(q_i(j), P)$
17. `check`$(q_i(j+1), P)$
18. **end if**
19. **end for**
20. **end for**
21. **return** P

Note that A is an $N \times N$ probability matrix, in which

$A[d_i][d_j]$ represents the access probability of the case d_i is requested just before d_j is requested. The queries are processed according to the access probability in decreasing order and for each query q_i is evaluated by two consecutive items $q_i(j)$ and $q_i(j+1)$ at each stage, $1 \leq j \leq n_{q_i} - 1$. When one of the two items is scheduled, our algorithm will re-check if there still exists better position for it in this moment. The procedure `insertAfter`(d, i, P) allocates the unscheduled data item d to the placement that has the minimum Δ from position i to d. Likewise, the procedure `insertBefore`(d, i, P) allocates the unscheduled data item d to the position with minimum Δ from d to placement i. The procedure `check`(d, P) exchanges d to some free position. The procedure `evaluate`(d, i) calculates the total access time when d is in the position i.

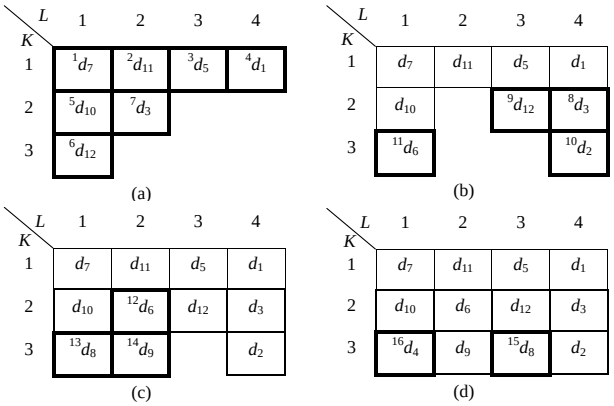

Fig. 4. An example of scheduling.

Consider the example of eight queries as shown in Table III. Assume that the number of channels K is 3. Since there are 12 data items, the length of broadcast cycle $L = 12/3 = 4$. A is an $N \times N$ probability matrix. The element $A[d_i][d_j]$ records the probability of the case that d_i requested before d_j. For example, the consecutive items d_7 and d_{11} only are needed in q_1, then $A[7][11] = 0.3679$. On the other hand, two consecutive items d_{12} and d_3 are needed in q_2 and q_4, $A[12][3] = 0.1839 + 0.0919 = 0.2758$. Thus we allocate q_1 and q_2 as shown in Fig. 4(a). When we consider d_3 and d_{10} in query q_2, `check`(d_3, P) finds better placement for d_3 and exchanges it. Similarly, query q_3 is arranged as shown in Fig.4(b) after the steps of d_{12} and d_2. Then query q_3 is adjusted as shown in Fig.4(c) after the steps of d_6 and d_2. When d_4 is allocated in P, all data items are scheduled as shown in Fig. 4(d) and the algorithm stops.

B. The case that the query distribution is unknown

In real applications, the exact query distribution is hard to obtain because the clients may sent queries or new clients might join in the system in any time. The probability matrix can not be known in advance, so we apply a frequency matrix, denoted by F, to record the history of data items in queries. When the sever receives a new query, it updates the element $F[d_i][d_j]$ by adding 1 after a broadcast cycle is finished, and then the server re-schedules and uses new broadcast program in the next broadcast cycle. The related algorithm is described in Algorithm 2.

Algorithm 2 `schedule2`

Require: Q: a query set with $M = |Q|$
q_i: a new unscheduled query
D: a data item set
K: the number of channels
L: the number of positions in each channel
F: the corresponding frequency matrix
Ensure: The broadcast program matrix P
1. add q_i into the unscheduled query set Q
2. **for** $k = 1$ to n_i **do**
3. $F[q_i(k)][q_i(k+1)] = F[q_i(k)][q_i(k+1)] + 1$
4. **if** $q_i(k) \notin D$ **then**
5. add $q_i(k)$ into D
6. **end if**
7. **end for**
8. $N = |D|$
9. `resize`(N, K, L, P)
10. `schedule1`(Q, D, K, F)
11. **return** P

Since the queries are not static, the size of broadcast program P is not constant. Thus we need to adjust the size of P. If we set L as large as possible, it will take too much time to check free placements for locating a better solution. On the other hand, if L is too small, there are fewer free placements to examine such that the access time will increase. The `resize` procedure is suggested in Algorithm 3 to alter the dimension of P dynamically for containing some more free placements such that data items can be re-allocated. When the free placements are not enough, K and L are added by 1 simultaneously. Thus, when the systems opens a new channel to schedule, every position in this channel is available to be assigned.

Algorithm 3 `resize`(N, K, L, P)

Require: N: the number of scheduled items
K: the number of channels
L: the number of positions in each channel
P: the old broadcast program matrix
Ensure: The new broadcast program matrix P
1. **while** $K * L \leq N +$(the average length of queries)$/2$ **do**
2. $L++$
3. $K++$
4. **if** $L > maxL$ **then**
5. $L = maxL$
6. **end if**
7. **if** $K > maxK$ **then**
8. $K = maxK$
9. **end if**
10. **end while**
11. P is a $K \times L$ matrix
12. **return** P

IV. Simulation

This section presents the experimental results of `schedule1` and `schedule2`. Two measurements are considered: access time and execution time. The test sample of ordered queries are generated by Zipf distribution [27], $f_i = (1/i)^\theta / \sum_{j=1}^{N}(1/j)^\theta$ where $1 \leq i \leq M$ and θ is skewness parameter. When $\theta = 0$, the distribution is uniformly distribution. When $\theta = 1$, the top 20 percent of queries request 80 percent of data items. In busy web sites, the traffic obeys the distribution when θ is large than 1 [22].

A. The case that query distribution is known

The related parameters when the query distribution is known are listed in Table IV. The results are also compared to OLS and MULS [11]. Fig. 5 and Fig. 6 show that `schedule1` the average access time is lowest while the execution time is between those of OLS and MULS. This reveals that `schedule1` takes reasonable amount of time to improve the access time. In fact, MULS may suffer the local minimum problem since it just checks and exchanges possible solutions pair by pair. From Fig. 7, we obtain that the average access time decreases when θ increases. This is because these algorithms allocate the higher probability queries first. Moreover the average access time of `schedule1` is very close to that of MULS, and it is lowest when θ is large than 0.8.

TABLE IV
PARAMETERS USED WHEN THE QUERY DISTRIBUTION IS KNOWN.

Parameters	Value	Range
N	240	220 $\sim$ 380
K	12	4 $\sim$ 12
M	100	100
θ	1.0	0.0 $\sim$ 1.2
average query length	100	10 $\sim$ 120

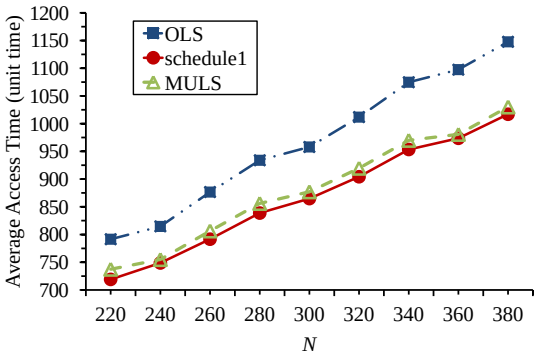

Fig. 5. Average access time with N varied when the query distribution is known.

B. The case that the query distribution is unknown

Table V lists the values of all parameters used when the query distribution is unknown. Since MULS needs more execution time, we just modified the original OLS to be applied in this scenario. From Fig. 8, it is shown that the average access time increases when N increases and `schedule2` algorithm is more efficient than the modified OLS version. Similarly, even to the execution time, `schedule2` takes reasonable amount of time to improve the access time.

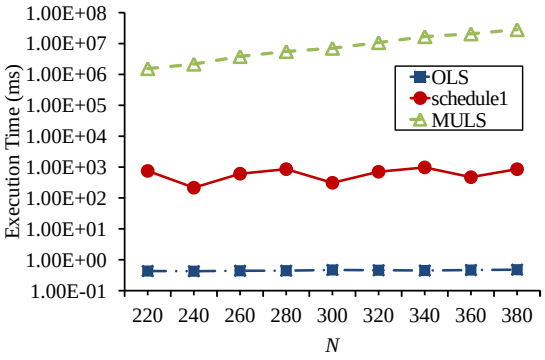

Fig. 6. Execution time with N varied when the query distribution is known.

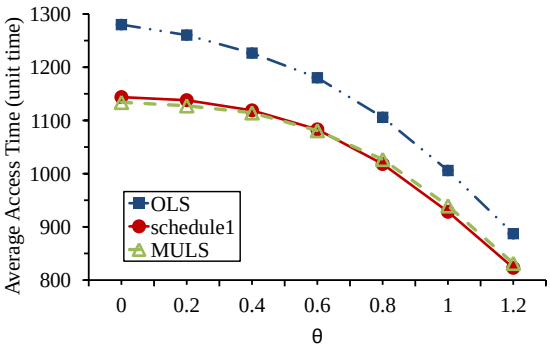

Fig. 7. Average access time with θ varied when the query distribution is known.

TABLE V
PARAMETERS USED WHEN THE QUERY DISTRIBUTION IS UNKNOWN.

Parameters	Value	Range
N	200	100 $\sim$ 400
K	10	10
M	50	50
θ	1.0	0.0 $\sim$ 1.2
average query length	100	10
query rate per unit time	10	10

V. Conclusion

Data broadcasting is an efficient and scalable method to disseminate information to a large amount of mobile clients. In this paper, we studied the problem of data allocation for ordered queries on multiple broadcast channels and proposed an efficient algorithm based on a local search strategy. The experimental results show that our algorithm outperforms the existing algorithms when the query distribution is known. Moreover, our algorithm can be applied efficiently when the query distribution is unknown, and can reduce the average access time as well.

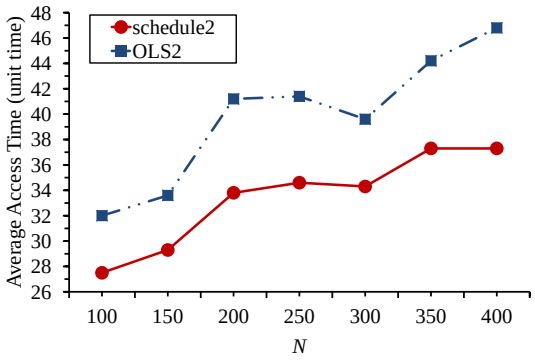

Fig. 8. The average access time with N varied when the query distribution is unknown.

REFERENCES

[1] S. Acharya, R. Alonso, M. Franklin and S. Zdonik, "Broadcast disks: Data management for asymmetric communication environments," *Proceedings of the ACM SIGMOD Conference*, 1995, pp. 199-210.

[2] S. Acharya, M. Franklin and S. Zdonik, "Disseminating updates on broadcast disks," *Proceedings of the 22th International Conference on Very Large Data Bases*, 1996, pp. 354-365.

[3] S. Acharya, M. Franklin and S. Zdonik, "Balancing push and pull for data broadcast," *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, vol. 26, no. 2, 1997, pp. 183-194.

[4] Y.C. Chehadeh, A.R. Hurson and M. Kavehrad, "Object organization on a single broadcast channel in the mobile computing environment," *Multimedia Tools and Applications*, vol. 9, no. 1, 1999, pp. 69-94.

[5] Y.D. Chung and M.H. Kim, "QEM: A scheduling method for wireless broadcast data," *Proceedings of the Sixth International Conference on Database Systems for Advanced Applications*, 1999, pp. 135-142.

[6] Y.D. Chung and M.H. Kim, "Effective data placement for wireless broadcast," *Distributed and Parallel Databases*, vol. 9, no. 2, 2001, pp. 133-150.

[7] S.M.S. Google, "Google Short Message Service(SMS)," 2005; http://www.google.com/sms/.

[8] J.L. Huang and M.S. Chen, "Dependent data broadcasting for unordered queries in a multiple channel mobile environment," *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, no. 9, 2004, pp. 1143-1156.

[9] J.L. Huang and M.S. Chen, "Exploiting replication on dependent data allocation for ordered queries over multiple broadcast channels," *Wireless Networks*, published online.

[10] J.L. Huang, M.S. Chen and W.C. Peng, "Broadcasting dependent data for ordered queries without replication in a multi-channel mobile environment," *Proceedings - International Conference on Data Engineering*, 2003, pp. 692-694.

[11] H.P. Hung and M.S. Chen, "MULS: A general framework of providing multilevel service quality in sequential data broadcasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 10, 2007, pp. 1433-1447.

[12] H.P. Hung, J.W. Huang, J.L. Huang and M.S. Chen, "Scheduling dependent items in data broadcasting environments," *Proceedings of the 2006 ACM symposium on Applied computing*, 2006.

[13] G. Lee and S.C. Lo, "Broadcast data allocation for efficient access of multiple data items in mobile environments," *Mobile Networks and Applications*, vol. 8, no. 4, 2003, pp. 365-375.

[14] G. Lee, S.C. Lo and A.L.P. Chen, "Data allocation on wireless broadcast channels for efficient query processing," *IEEE Transactions on Computers*, vol. 51, no. 10, 2002, pp. 1237-1252.

[15] K.C.K. Lee, W.C. Lee and S. Madria, "Pervasive data access in wireless and mobile computing environments," *Wireless Communications and Mobile Computing*, vol. 8, no. 1, 2008, pp. 25-44.

[16] H.V. Leong and A. Si, "Data broadcasting strategies over multiple unreliable wireless channels," *International Conference on Information and Knowledge Management, Proceedings*, 1995, pp. 96-104.

[17] K.F. Lin and C.M. Liu, "Broadcasting dependent data with minimized access latency in a multi-channel environment," *IWCMC 2006 - Proceedings of the 2006 International Wireless Communications and Mobile Computing Conference*, pp. 809-814.

[18] K.F. Lin and C.M. Liu, "Schedules with minimized access latency for disseminating dependent information on multiple channels," *Proceedings -2006 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pp. 344-351.

[19] C.M. Liu and K.F. Lin, "Efficient scheduling algorithms for disseminating dependent data in wireless mobile environments," *Proceedings - 2005 IEEE International Conference on Wireless Networks, Communications and Mobile Computing*, 2005.

[20] S.C. Lo and A.L.P. Chen, "Optimal index and data allocation in multiple broadcast channels," *Proceedings - International Conference on Data Engineering*, 2000, pp. 293-302.

[21] S.P.O.T. Microsoft, "Microsoft DirectBand Network," 2006; http://www.msndirect.com/.

[22] V. Padmanabhan and L. Qiu, "The content and access dynamics of a busy web site: findings and implications," *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2000.

[23] W.C. Peng and M.S. Chen, "Efficient channel allocation tree generation for data broadcasting in a mobile computing environment," *Wireless Networks*, vol. 9, no. 2, 2003, pp. 117-129.

[24] K. Prabhakara, K.A. Hua and J. Oh, "Multi-level multi-channel air cache designs for broadcasting in a mobile environment," *Proceedings - International Conference on Data Engineering*, 2000, pp. 167-176.

[25] A. Si and H. Va Leong, "Query optimization for broadcast database," *Data and Knowledge Engineering*, vol. 29, no. 3, 1999, pp. 351-380.

[26] K. Stathatos, N. Roussopoulos and J.S. Baras, "Adaptive data broadcast in hybrid networks," *Proceedings of the 23rd International Conference on Very Large Data Bases*, 1997, pp. 326-335.

[27] G.K. Zipf, *Human Behavior and the Principle of Least Effort: An Introduction to Human ecology*, Addison-Wesley Press, Cambridge, MA., 1949.