

Energy-Efficient Scheduling Algorithms of Object Retrieval on Indexed Parallel Broadcast Channels *

Bingjun Sun, Ali R. Hurson, and John Hannan
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802-6106
{bsun, hurson, hannan}@cse.psu.edu

Abstract

With the goal of providing “timely and reliable” access to information in a mobile computing environment, mobile units and the wireless medium operate under constraints on energy, bandwidth, and connectivity. Among these limitations, power limitation of mobile units is one of the key issues.

In a mobile computing environment, broadcasting has proved to be an effective method to distribute public data. Efficient methods for allocating and retrieving objects on parallel indexed broadcast channels have been proposed to manage power consumption and access latency. Employment of parallel channels also brings out the notion of conflicts. To minimize the effect of conflicts on both access latency and power consumption, one has to develop schemes to schedule access to the objects that minimizes the number of passes over the parallel channels.

This work extends our past efforts and proposes two new scheduling algorithms that can find the minimum number of passes and inside channel switches. The simulation results show that the proposed scheduling algorithms relative to our previous work have a great impact on energy consumption and access latency. The proposed scheduling algorithms are simulated and results are presented.

Keywords: Mobile computing, Energy-aware scheduling, Parallel broadcast channels, object retrieval

1. Introduction

Multidatabases (MDB) are capable of allowing timely and reliable global access to large amounts of heterogeneous or homogeneous data sources in an environment that is characterized as “sometime, somewhere” [2]. However, with the rapid expansion of technology, the concept of *mobility* has introduced additional complexities to MDB designers.

Global data sources can be classified as [10]: *private data*, *public data*, and *shared data*. Broadcasting is an efficient method to make public data available to a large number of users. Previous studies have shown that retrieving data objects from the air channel(s) is a major source of power consumption in mobile units [4, 12, 13]. To manage power consumption, one needs to develop power-aware protocols that run hardware units in different operational modes [3, 5-7]. The literature has suggested several schemes to allocate/retrieve data objects to/from single or parallel broadcast air channel(s) [3-5]. In general, these schemes are based on indexing techniques to reduce the active time of mobile units. Furthermore, when parallel channels are employed, due to the conflicts, proper protocols should be developed to schedule access to the indexed parallel channels to reduce power consumption and response time by reducing the number of broadcast passes and channel switches [12]. Within the scope of indexed parallel broadcast channels, heuristic rules were applied to generate access patterns, during the broadcast cycles, to reduce both the number of passes and channel switching frequency [9, 12-13]. However, these algorithms do not necessarily generate optimal access latency and power consumption in all cases.

This work extends the scope of our earlier investigation. It proposes and examines two new scheduling algorithms that generate access patterns with minimum response time and energy consumption when retrieving objects from indexed parallel broadcast channels. The rest of this paper is organized as follows. Section 2 provides background information related to the broadcasting over parallel channels. Section 3 reviews the general access protocol of object retrieval from indexed parallel air channels, and then proposes several scheduling algorithms. Section 4 presents and analyzes the simulation results. Finally, in section 5, the conclusions are drawn, and future research directions are discussed.

* This work in part has been supported by The Office of Naval Research under the contract N00014-02-1-0282 and National Science Foundation under the contract IIS-0324835.

2. Background

The multidatabase concept was proposed to manage, share, and integrate the existing “islands of information” into a coherent system in a sometimes, somewhere environment. Recent advances in technology have made mobile computing a reality. However, in a wireless mobile computing environment, one should take technological constraints into consideration.

2.1 Wireless Computing

The wireless computing architecture consists of mobile hosts and fixed hosts. The network servers communicate with mobile units through mobile support stations (MSSs). In this environment two types of services are available [7, 9]:

- **Interactive/on-demand service:** The client requests a piece of data on the uplink channel, and the server responds by sending the result to the client - the channels are bi-directional and asymmetric.
- **Broadcasting service:** Periodic broadcasting of data on the channel - the channels are unidirectional from servers to mobile units.

2.2 MDAS

Mobile Data Access Systems (MDAS), as an extension to MDDBS, was proposed to support wireless communication, with the goal of accessing heterogeneous and autonomous data sources anytime and anywhere [10]. The current limited bandwidth in a wireless environment places an obstacle on the rate and the amount of conventional two-way communication in a MDAS [3].

2.3 Data Broadcasting

Wireless data broadcasting is an efficient method to make public data available to the users while overcoming the technological limitations of the wireless network. In addition, data broadcasting is scalable and the air channel(s) can be assumed as the extended memory of mobile devices [1]. Data on a broadcast channel are read-only and must be accessed sequentially. Data can be broadcast either on a single channel or parallel channels. This work assumes that all parallel channels have the same transfer rate, and data broadcasting will be done in a cyclic manner.

2.4 Allocation and Retrieval Methods on Parallel Air Channels

Within the scope of broadcasting, the indexing technique can be used to reduce power consumption. In addition, broadcasting along parallel channels can be used to reduce the broadcast length and hence to reduce the access latency. However, employment of parallel channels introduces conflicts which could

enforce multiple passes over the parallel air channels. This increases the power consumption and access latency. Therefore, it would be desirable to schedule the object retrieval during each broadcast cycle with the goal of reducing the number of passes [9, 12].

2.4.1 Indexing Techniques on Air Channels

With indexing techniques, the mobile unit can predict the arrival time of requested objects in order to switch into an energy-saving mode [3]. The advantages of indexing schemes come at the expense of computational overhead and increased broadcast length. Several indexing techniques, such as signature and tree based, have been addressed in the literature [4-9, 12-13]. This work employs the tree-based index techniques.

An index tree is the auxiliary information representing one or several data attributes pointing to the location of a data frame sharing the same common attribute value(s). Nodes at the lowest level of the index tree point to the locations of the data frames on the broadcast. Previous works have studied different indexing allocation schemes (namely, *distributed indexing* and *(1, m) indexing*) [6-8].

Single-class indexing and *hierarchical* indexing in single and parallel broadcast channel(s) have been studied in detail. The simulation results showed that the application of an index on the single broadcast air channel can reduce energy consumption drastically with a reasonable increase in response time due to a longer broadcast [3]. In parallel broadcast channels, for the case of single-class scheme, there are several possible layouts to organize index and data. Without loss of generality, this work will employ a layout that is depicted in Figure 1 because of its simplicity and clarity.

2.4.2 Object Retrieval on Parallel Channels

Within the scope of parallel broadcast channels, the mobile unit can switch between different channels and download objects. However, usually not all the requested objects can be accessed in one pass due to the conflicts among requested objects when broadcasting simultaneously and/or during channel switching periods. A general access protocol to retrieve data objects from parallel channels should minimize power consumption and response time [12]. The protocol involves the following steps:

- 1) **Initial probe:** The client tunes into the broadcast channel to determine when the next index will be broadcast.
- 2) **Search:** The client accesses the index and determines the offsets of requested objects.
- 3) **Compute:** The client generates the access patterns for the requested objects using a scheduling algorithm.

- 4) **Retrieve:** The client, in an active mode, tunes into one of the channels and downloads the required data objects and changes to doze mode if needed.

I _H	D/c
F/s	D/c
F/s	D/c
F/s	D/c

F/s Free space
I_H Hierarchical Index
D/c Data on each channel

Figure 1: Broadcast Layout

During the *compute* step, scheduling algorithms can attempt to minimize the number of passes and the number of channel switches, at a reasonable overhead. There are two kinds of channel switches: Inside switches and outside switches. Inside channel switches are those occurring during a broadcast cycle. Outside switches are channel switches which occur between two successive broadcast cycles.

Several heuristic-based algorithms have been proposed to schedule object retrieval during a broadcast cycle. The solution to the Traveling Salesman Problem, the *Next Object*, and row scan heuristics were used to access objects from the air channels [9]. As reported in [12], three rules were employed to generate the access patterns.

3. Energy-Efficient Scheduling of Object Retrieval

3.1 Object Retrieval on Parallel Broadcast Channels

The literature has identified the Access Time, the Tune-in Time, and the number of Channel Switches as the performance metrics to measure the effectiveness of retrieval schemes [5, 7]. We use response time and energy consumption as the performance metrics to evaluate the effectiveness of our proposed methods. The overall power consumption is computed as (assuming that the power consumption for channel switching is 10% of the power consumed in active mode):

$$\text{Energy Consumption} = (\text{Access Time} - \text{Tune-in Time}) * \text{DozeModePower} + (\text{Tune-in Time}) * \text{ActiveModePower} + \text{TheNumberOfSwitching} * 10\% * \text{ActiveModePower}.$$

3.2 Scheduling Algorithms on Indexed Parallel Channels

3.2.1 Problem Analysis

In this work, a two-dimensional array of $N \times M$ is used to represent a parallel broadcast, where N is the number of channels and M is the number of pages on

a channel. In this array, each cell C_{ij} , $1 \leq i \leq N$, $1 \leq j \leq M$, represents a page. Without loss of generality, we assume objects are not fragmented across adjacent pages, and if we request any object in a page, we will retrieve the whole page. A row of cells represents a channel on the broadcast, while a column of cells represents pages transmitted at the same time on parallel channels. For a query requesting a set of objects S , if the cell C_{ij} has a requested object then $C_{ij} \in S$. Finally, a query requests K objects O_k , $1 \leq k \leq K$. Based on the aforementioned assumptions we have the following definitions with respect to a given query (Figures 2 and 3 are intended to clarify these definitions).

Definition 1: Empty Columns: A column without any requested objects.

Definition 2: Sparse Columns: A column containing requested objects without any conflicts.

Definition 3: Dense Columns: A column that is neither *Empty* nor *Sparse* is *Dense*. With respect to a user request, the minimum required number of passes depends on dense columns.

We define R_j to be the set of rows we need to scan in column j based on objects in columns j through $j+OPR-1$ (OPR stands for overlap page region). This naturally leads to the definition of the cut at column j .

Definition 4: Let $R_j = \{i \mid \exists j', j \leq j' \leq j+OPR-1, \text{ such that } C_{ij'} \in S\}$. Then, the *cut* at column j is $|R_j|$.

Definition 5: The *maximum cut*, MAX_cut , is $\max\{cut_j \mid 1 \leq j \leq N\}$

Definition 6: The *total requested channels*, MAX_total , is $|\{i \mid \exists j \text{ such that } C_{ij} \in S\}|$.

We can conclude $MAX_cut = \text{minimum number of passes} = MAX_total$.

We define some useful sets of cells:

Definition 7: The set of *Empty Cells*, E , is $\{C_{ij} \mid C_{ij} \notin S\}$.

Definition 8: The set of *Restrictive Cells*, R , is $\{C_{ij} \mid C_{ij} \notin S \wedge (C_{i,j+1} \in S \vee C_{i,j-1} \in S)\}$.

Definition 9: The set of *Free Cells*, F , is $\{C_{ij} \mid C_{ij} \in E \wedge C_{ij} \notin R\}$.

An important concept in our algorithms is the empty block which is a contiguous sequence of empty cells in one row.

Definition 10: An *Empty Block*, B_{ij} , is the largest set of contiguous empty cells in row i starting in column j .

Given a set of empty blocks all starting in the same column, a *rightmost empty block* is a block in the set that is not smaller than any other block in the

set, i.e., no other empty block extends farther to the right of it.

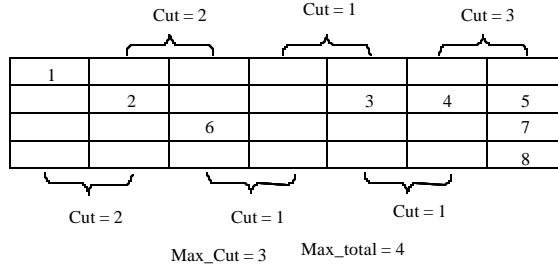


Figure 2: cut, MAX_cut and MAX_total

Definition 11: Overlapped Empty Blocks: Two empty blocks B and B' overlap each other, iff they contain a cell from some column, i.e., $\exists j$, such that $C_{i,j} \in B \wedge C_{i',j} \in B' \wedge i \neq i'$.

Definition 12: The set of *Obstacles*, O , is $\{C_{i,j} \mid C_{i,j} \in S \wedge C_{i,j-1} \in E \wedge C_{i,j-2} \in E\}$.

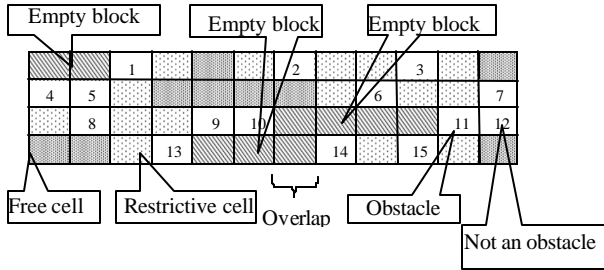


Figure 3: Free cells, restrictive cells, overlapped empty blocks, and obstacles

3.2.2 Parallel Object Scan

In this presentation, graphically, lines (access lines) are used to represent the access patterns in different passes. Parallel Object Scan (POS) is a scheduling algorithm that uses MAX_cut passes to scan all requested objects on a broadcast. Starting in the first column the algorithm simultaneously (in parallel) constructs MAX_cut lines, column by column, working left to right. The algorithm attempts to use access lines to visit all the requested objects in the next column with the fewest switches from the previous column. POS observes which cell $C_{i,j}$ with a line has a less chance to have a requested object in the future on the same channel and uses this line to make an inside switch when necessary. This strategy can minimize the number of inside switches.

Definition 13: A Line l is a vector of length M representing one pass through the matrix. The value $l[j]$ is the channel (row #) that the line reads at time (column) j .

We define $freeright(i,j)$ to be the rightmost column of the *Empty block* containing $C_{ij} \in E$.

Definition 14: Let $freeright(i,j)$ be the largest j' such that for all $k, j \leq k \leq j'$, $C_{ik} \in E$.

We define $next(j,L)$ to be the line in L that reads from the channel with the rightmost empty block in column j .

Definition 15: Let $next(j,L)$ be a line $l \in L$ such that $freeright(l[j],j) = \max\{freeright(l'[j],j) \mid l' \in L\}$.

We define $First(i)$ to be the column in which a requested object first appears in row i .

Definition 16: Let $First(i)$ be the smallest j such that for all $j', 1 \leq j' < j$, $C_{ij'} \notin S$.

Let $Start(m)$ be rows $\{i_1, \dots, i_m\}$ such that for any row $i' \notin \{i_1, \dots, i_m\}$ $First(i) \leq First(i')$ for all $i \in \{i_1, \dots, i_m\}$. In other words, $Start(m)$ are rows that contain objects before any other rows. These rows will be the starting point for our MAX_cut lines, as we obviously do not want to switch a line before it has read any objects on a channel.

The POS algorithm is shown below:

POS Algorithm:

1. if ($MAX_total = MAX_cut$)
2. use Row Scan;
3. else
4. Let $m = MAX_cut$
5. Create m lines, l_1, \dots, l_m
6. $Lines = \{l_1, \dots, l_m\}$ /* Set containing all the lines */
7. $l_1[1], \dots, l_m[1] = Start(m)$ /* Initialize lines */
8. for $j = 2$ to M do {
9. $L = Lines$ /* All the lines */
10. $A = R_j$ /* Required Rows: $|A| \leq |L|$ */
11. foreach $l \in L$ /* check for no switch, line reads an object */
12. if $l[j] \in A$ then $l[j] = l[j-1]$; $A = A - \{l[j]\}$; $L = L - \{l\}$
13. foreach $i \in A$ /* remaining objects, inside switch */
14. $l = next(j,L)$
15. $l[j] = i$
16. $L = L - \{l\}$
17. foreach $l \in L$ /* no object to read, no switch */
18. $l[j] = l[j-1]$

The POS algorithm runs in $O(N^2M)$ time. Observe that all of the required sets and functions except $next$ (e.g., R_j , E , $freeright$, $First$, $Start$) can be computed in $O(NM)$ time. If we pre-compute these once, then any use of them during the algorithm

requires just constant time. The function *next* takes $O(N)$ time since the size of L (the number of lines) will be at most N . The algorithm consists of an outer loop iterated $M-1$ times. Inside are three inner loops each iterated at most N times (since the sizes of L and A are bounded by N). The first and third inner loop bodies take constant time while the second inner loop body takes $O(N)$ time due to the call to *next*.

3.2.3 Serial Empty Scan

An alternative to POS is Serial Empty Scan (SES) which examines empty blocks instead of requested objects. The basic idea behind the algorithm is as follows. We construct ($MAX_total - MAX_cut$) paths that scan only empty blocks (empty paths). As we do this we also compress the requested objects into MAX_cut channels. Each of these resulting “logical” channels describes the sequence of requested objects that an access line reads. The action of compressing (copying objects from one channel to another) simulates a switch during a scan.

The SES algorithm is serial because it finds the ($MAX_total - MAX_cut$) empty paths one by one. The SES algorithm constructs an empty path by starting with the rightmost empty block $B_{i',l}$ in some row i' . The path starts in row i' at column 1 and continues to column k , where k is the length of $B_{i',l}$. Next it finds a rightmost overlapping empty block $B_{i'',k}$. The path continues on row i'' for the length of this empty block and so forth. Additionally, the SES algorithm begins to create a logical channel by moving all requested objects $C_{i',a}$, to $C_{i'',a}$, for $1 \leq a < k$. Finally it marks the empty cell $C_{i'',k}$ as *no obstacle*, indicating a physical switch occurs at this point in the logical channel. This end marker is used during the construction of subsequent empty paths. By convention, we also define that $\forall i', C_{i',M} = no\ obstacle$, iff $C_{i',M} \in E \wedge C_{i',M-1} \in E$. The SES algorithm repeats this construction of an empty path until it reaches the right end. During subsequent scans, when choosing an overlapping empty block, the SES algorithm chooses the rightmost block ending with the *no obstacle* marker, if one exists. If no such empty block exists, the algorithm simply chooses the rightmost empty block. After every scan, MAX_total is decremented by 1. The algorithm will terminate when $MAX_total = MAX_cut$, and all the requested objects have been moved into MAX_cut logical channels, on each of which the order of the requested objects gives the access pattern during each broadcast.

Since choosing the next empty block is a fundamental step in the SES algorithm we define it first. Given a set of row indices $Rows$ and a column j , *nextblock*($Rows, j$) returns the appropriate row i to

add to the empty path being constructed. To define this function we also define *rightmost*($Rows, j$) which returns the row index of the rightmost free block among B_{ij} for $i \in Rows$.

$rightmost(Rows, j) = i$ s.t. $i \in Rows$ & $freeright(i, j) = \max(freeright(i', j) \mid i' \in Rows)$

nextblock($Rows, j$) =

let EBs = { $i \mid i \in Rows$ & $C_{ij} \in E$ } /* Empty Blocks */

let OBs = { $i \mid i \in EBs$ & $k = freeright(i, j)$ & $C_{ik} = no\ obstacle$ }

if OBs $\neq \{ \}$ then return(*rightmost*(OBs, j)) else return(*rightmost*(EBs, j))

In the SES algorithm, an obstacle on a broadcast may result in an inside switch if the empty block on the left of it is scanned. Those obstacles whose empty blocks on the left are not chosen to scan will not result in a switch. Cells that are not obstacles will not generate inside switches. It should be noted that if the algorithm moves requested objects from channel A to channel B , a switch is indicated, but if they are subsequently moved from channel B to channel C , it is equal to moving from channel A to channel C , and still only one switch occurs. Thus, the strategy of choosing blocks marked no obstacle (in the function *nextblock*) minimizes the number of inside switches. Choosing the rightmost empty block, at each step also contributes to minimizing the number of inside switches of a scan.

SES Algorithm:

if ($MAX_total = MAX_cut$)

use Row Scan;

else

$Rows = \{1, \dots, N\}$ /* Rows we can compress */

repeat until ($MAX_total = MAX_cut$) {

$j = 1$;

$i = nextblock(Rows, j)$

$k = freeright(i, j)$ /* free block is C_{ij} to C_{ik} */

repeat until $k=M$ {

$i' = nextblock(Rows, j)$

$k = freeright(i', j)$

$C_{ia} = C_{i'a}$ for $1 \leq a < k$ /* compress channels */

$C_{i'k} = no\ obstacle$ /* indicate a switch occurs */

$i = i'$

$j = k$

}

$MAX_total = MAX_total - 1$

$Rows = Rows - \{i\}$ /* delete the empty row */

}

The SES algorithm has complexity $O(N^2M)$. The functions *rightmost* and *nextblock* are both bounded by the number of rows, hence they are both $O(N)$. The SES algorithm's outer loop iterates at most N times and its inner loop iterates at most M times. Finally, the body of the inner loop takes $O(N)$ times due to the call to *nextblock*. Consequently, the inner loop has complexity $O(NM)$ and so does the body of the outer loop.

4. Simulation Results

4.1 Simulation Design

A simulator was developed to verify the behavior of the proposed algorithms and measure their effectiveness against the tree based indexing algorithm proposed in [12, 13]. The simulator assumes a hierarchical inheritance-based indexing as the underlying object organization on the parallel air channels for a retrieval protocol. For the sake of simplicity, the simulator simulates the "Layout 1" configuration as depicted in Figure 1. Finally, to compensate for the execution time of each algorithm, a delay gap of one page is inserted between the index and data objects to avoid missing any requested objects in the first broadcast pass. For each simulation run, user requests are generated requesting K random objects on the broadcast.

The NASDAQ [12] database with 4290 securities is used as the source data for the objects on the broadcast. The simulator assumes some physical parameters as input. These parameters are summarized in Table 1. It should be noted that, for every simulated configuration, the simulator is run 1000 times, and the average number of every estimated performance metric is calculated and represented in this paper.

Table 1: Input parameters to the simulator

Parameter	Value
Number of Objects	4290
Number of Channels	1-16
Size of Air Page	512 Bytes
Broadcast Data Rate	1 Mbit/sec
Power Consumption	
(Active Mode)	130 mW
(Doze Mode)	6.6 mW
(Channel Switch)	13 mW

4.2 Simulation Results

Conceptually, the POS and SES algorithm should generate the same number of passes and inside switches regardless of the simulated configuration. The simulator ran 1000 times for 80 different configurations, and the results verified this point.

4.2.1 Number of Passes

Figure 4 shows the number of passes for the POS, SES, and tree-based algorithms (abbreviated as "Tree" in the figures). As expected, regardless of the number of objects requested and the physical configuration of the simulated environment, the POS, and SES algorithms generate the same number of passes.

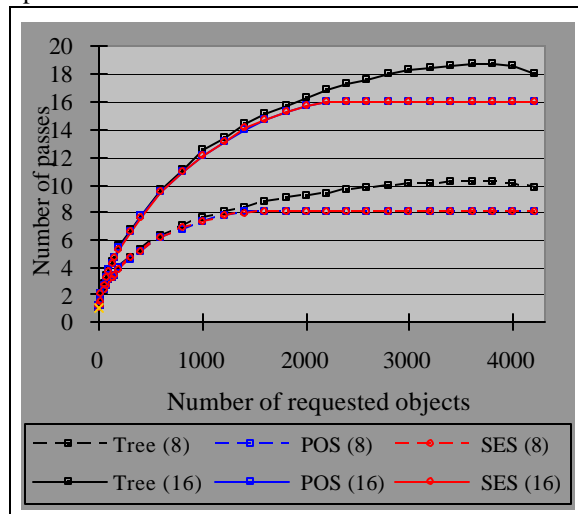


Figure 4: Number of passes

When K is relatively small, the tree-based algorithm is as efficient as the POS and SES algorithms. However, as K grows larger, the tree-based algorithm became relatively less efficient. Interestingly, relative to the number of air channels, after a threshold value, the tree-based algorithm requires more passes than *Row Scan* algorithm. Increasing K and/or N increases the number of passes required to pull requested objects from the parallel air channels. An increase in N would increase the probability of conflicts and, consequently, results in a higher *MAX_cut* value. This would increase the number of required passes over the air channels. As K approaches $((M * (N - 1)) + 1)$, *MAX_cut* approaches N . At this threshold point, the algorithm behaves as *Row Scan* algorithm since it would be the most cost efficient algorithm to use.

4.2.2 Number of Channel Switches

Figure 5 shows the number of channel switches for different scheduling algorithms. As expected, the POS and SES algorithm require almost the same number of switches regardless of the number of channels and/or the number of objects requested. This is due to the fact that the POS and SES algorithms can find the optimal solution of inside switches. Compared to the POS and SES algorithm, the tree-based algorithm requires more channel switches, especially as K grows larger.

As N and/or K increases, more inside switches are needed since, more requested objects are distributed over different channels. When K increases beyond some threshold point, the number of channel switches begins to decrease. This is because the number of passes approaching to N . As K approaches $((M * (N - 1)) + 1)$, no inside switches are required, since, as noted earlier, at this threshold point the *Row Scan* algorithm would be applied which requires no inside channel switches.

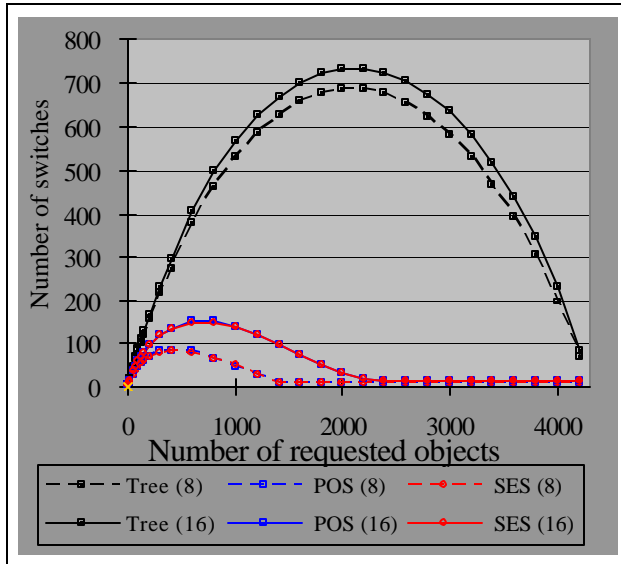


Figure 5: Number of switches

4.2.3 Cost of Algorithms

As noted earlier, it was one of our objectives to develop a scheduling algorithm that generates access patterns with reasonable overhead cost. As noted before, the cost of the tree-based algorithm is $O(K^2)$, much more than those of the proposed algorithms, especially when a large number of objects is requested. The cost of the POS and SES algorithms is $O(N^2M)$ in the worst case. The cost of POS and SES algorithms linearly increases when a relatively small number of objects are requested. As the number of requested objects increases, the cost of the algorithms decreases because more and more cases will use the *Row Scan* algorithm. After the threshold point, it is constant since the *Row Scan* algorithm is applied.

4.2.4 Response Time

Figure 6 shows the response time of different scheduling algorithms. Response time is directly related to the number of passes. The response time of the POS and SES algorithm is almost the same, while the tree-based algorithm usually requires a longer response time, especially when K is large. Interestingly, we observed that for small K , the POS and SES algorithm have a little longer response time than the tree-based algorithm. This is due to the

heuristics employed by the tree-based algorithm - it retrieves as many objects as possible in the earlier passes.

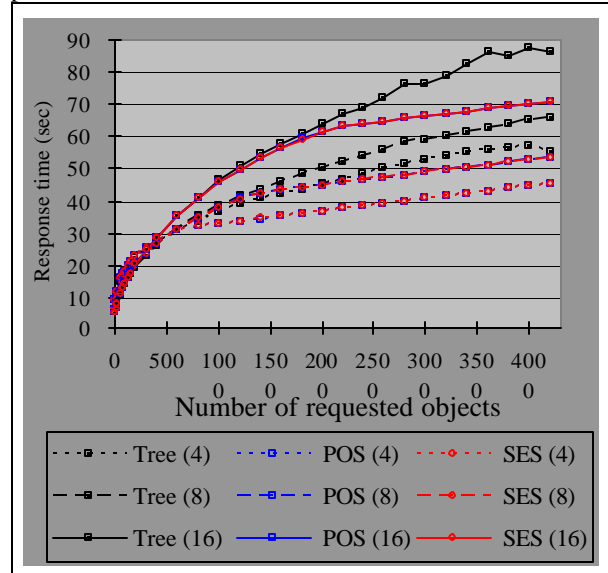


Figure 6: Response time

4.2.5 Energy Consumption

Figure 7 shows energy consumption of different scheduling algorithms. The total energy consumption is determined based on the consumed energy: in active mode, in doze mode, and in channel switching phase. Obviously, for different scheduling algorithms, the consumed energy in active mode is almost the same. The energy consumed in doze mode is not much different for different scheduling algorithms. As a result, one can conclude that the energy consumed in switching phase is the dominating factor that distinguishes different scheduling algorithms based on this performance metric. As a result, the shapes of the curves in figure 8 are similar to those of Figure 5.

4.2.6 Trade-off between Broadcast Passes and Channel Switches

There is a trade-off between the response time and the number of channel switches (energy consumption). To show this we ran the simulator for a fixed configuration, i.e., the same requested objects and the same number of channels, and observed the number of channel switches and the corresponding response time. The following observation was made: more passes result in longer response time and fewer channel switches. On the other hand, the energy consumption depends on the number of passes and the number of channel switches. In addition, the number of objects requested plays an important role in defining the relationship between energy consumption and the number of passes. For example,

for a relatively small number of requested objects, a smaller number of passes consumes less energy and for relatively a large number of requested objects, more passes over the parallel channels consume less energy. Between these two boundaries, a medial number of passes results in the least amount of energy consumption.

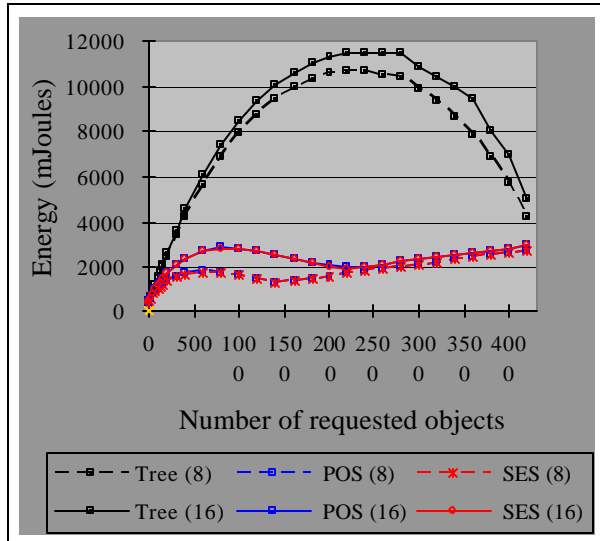


Figure 7: Energy consumption

5. Conclusions and Future Directions

Two new scheduling algorithms, namely, the Parallel Object Scan (POS) and Serial Empty Scan (SES), were presented along with their time complexity. The proposed algorithms were simulated and compared against the tree based algorithm proposed in [12, 13] using performance metrics such as the energy consumption, the response time, and the number of channel switches. Simulation results showed that the proposed scheduling algorithms, for a user request, generate access patterns with the minimum number of passes and inside switches at a reasonable overhead cost.

This work can be extended in many directions, including the following:

- To reduce response time and energy consumption further, one can develop a scheme that caches the index in the mobile units locally.
- The simulator can be extended to find the channel layout of allocating the index and data objects that offers the best performance.
- User's profile can be used to bundle multiple potential queries together to reduce the overall power consumption.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric

Communication Environments," *ACM SIGMOD Conference on the Management of Data*, 1995, pp. 199-210.

- [2] Bright, M.W., Hurson, A.R. and Pakzad, S. "Automated Resolution of Semantic Heterogeneity in Multidatabases," *ACM Trans. on Database Systems*, 19(2):212-223, 1994.
- [3] Y.C. Chehadeh, A.R. Hurson, L.L. Miller, "Energy-Efficient Indexing on a Broadcast Channel in a Mobile Database Access System," *Conference on Information Technology: Coding and Computing*, 2000, pp. 368-374.
- [4] Y.C. Chehadeh, A. R. Hurson, D. Tavangarian, "Object Organization on Single and Parallel Broadcast Channel," *Proceedings of the International Conference on High Performance Computing*, 2001, pp. 163-169.
- [5] Q.L. Hu, and D.L. Lee, "A Hybrid Index Technique for Power Efficient Data Broadcast," *Distributed and Parallel Databases Journal*, 9(2):151-177, 2001.
- [6] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Data on Air: Organization and Access," *IEEE Transactions on parallel and distributed systems*, 9(3):353-372, 1997.
- [7] T. Imielinski, and H.F. Korth, "Mobile Computing," *Kluwer Academic Publishers*, 1996.
- [8] T. Imielinski, and B.R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Communications of the ACM*, 37(10):pp. 18-28, 1994.
- [9] J. Juran, A.R. Hurson, and N. Vijaykrishnan, "Data Organization and Retrieval on Parallel Air Channels: Performance and Energy Issues," *ACM Journal of WINET*, 10(2), 2004.
- [10] J.B. Lim, A.R. Hurson, "Heterogeneous Data Access in a Mobile Environment - Issues and Solutions," *Advances in Computers*, Vol. 48, pp.119-178.
- [11] Lim J.B., and Hurson A.R., "Transaction Processing in Mobile, Heterogeneous Database Systems," *IEEE Transactions on Knowledge and Data Engineering*, 14(6):1330-1346, 2002.
- [12] Munoz-Avila A., and Hurson A.R., "Energy-Efficient Objects Retrieval on Indexed Broadcast Parallel Channels," *Conference on Information Resource Management*, pp. 190-194, 2003.
- [13] Munoz-Avila A., and Hurson A.R., "Energy-Aware retrieval From Indexed Broadcast Parallel Channels," *Advanced Simulation Technology*, pp. 3-8, 2003.
- [14] E. Pitoura, and P.K. Chrysanthos, "Scalable Processing of Read-Only Transactions in Broadcast Push," *Conference on Distributed Computing Systems*, pp. 432-439, 1999.
- [15] K. Stathatos, N. Roussopoulos, and J.S. Baras, "Adaptive Data Broadcast in Hybrid Networks," *International Conference on Very Large Data Bases*, pp. 326-335, 1997.