

# Efficient Object Retrieval from Parallel Air Channels in the Presence of Replicated Objects

Padmapriya Ayyagari  
[payyagari@ist.psu.edu](mailto:payyagari@ist.psu.edu)

Prasenjit Mitra  
[pmitra@ist.psu.edu](mailto:pmitra@ist.psu.edu)

Ali Hurson  
[hurson@cse.psu.edu](mailto:hurson@cse.psu.edu)

The Pennsylvania State University

## Abstract

*A mobile device retrieving objects from parallel air channels has to optimize multiple objectives: (1) reduce the response time for object retrieval, and (2) reduce power consumption in the device. This multi-objective optimization becomes even more difficult when the same object is broadcast multiple times over the parallel air channels to increase availability. In this work, several heuristic algorithms - branch and bound, greedy, random, and first-choice have been proposed for efficient object retrieval from parallel air channels in the presence of replicated objects. Empirical evaluation of the algorithms shows that the greedy algorithm produces near-optimal-response time with minimum overhead in terms of energy consumption. The runtime of the greedy algorithm is significantly faster than that of the optimal branch and bound algorithm, and is similar to the first-choice and random algorithms.*

## 1. Introduction

Data dissemination in mobile environments is made possible by two popular techniques [3, 14] – *unicast* (data access using point-to-point connection between the mobile device (*the client*) and the server) and *broadcast* (data dissemination on air channels). In the broadcast paradigm, the onus is on the client *to tune in* to the air channel and retrieve data.

Much of the work in data dissemination, organization, and retrieval in mobile environments has focused on the broadcast paradigm [1, 2, 3, 12, 13, 14]. Broadcasting is especially suitable for mobile environments where issues such as limited connectivity, limited bandwidth, and limited power are of concern [3]. In addition, broadcasting makes efficient use of bandwidth, and supports high workloads as the system scales to an increasing number of users. Furthermore, since mobile devices come with limited storage, a broadcast channel, can, in some sense, be considered an “extended” storage medium for mobile devices [6]. Typically, broadcasting is used for disseminating read-only data such as weather information, stock market information, geographical information, etc.

To improve average response time, data is often broadcast on parallel air channels rather than on a single long channel [1]. (Also, it may not be possible to coalesce multiple physical channels into a single channel due to

bandwidth constraints [13, 14]). In addition, frequently accessed objects can be broadcast several times during each broadcast cycle (re-transmitted) to improve the average response time and the availability [9, 10, 11]. In this paper, we refer to an object that is broadcast multiple times on the parallel air channels during the same broadcast cycle, as a *replicated object*.

Current mobile technology allows a mobile unit to tune into only one air channel at a time. Thus, if the data required by a user is distributed over multiple parallel channels, the mobile unit must switch between these channels to retrieve the desired data. A mobile client may not be able to retrieve all the data during a single broadcast cycle (see section 2.2). Thus, the mobile unit must plan an *access schedule* to decide the order in which to retrieve the objects from the broadcast. *Retrieving data over multiple broadcast cycles (multiple passes) increases response time, while switching between channels leads to increased energy consumption.* Therefore, a good access schedule is one that minimizes:

- the number of passes made over the broadcast, and
- the number of switches between air channels.

Sun, et al. [1], addressed the problem of generating efficient access schedules for object retrieval from parallel air channels. However, they did not address the problem of efficient object retrieval in the presence of replicated objects. In this case, the main challenge lies in deciding which one of the many copies of a replicated object should the mobile client retrieve so as to optimize response time and energy consumption. We propose efficient heuristic algorithms that give near-optimal solutions in terms of response time and energy consumption.

The rest of the paper is organized as follows. Section 2, discusses related work. In Section 3, we introduce some basic ideas and terminology. Section 4 defines the problem and Section 5 shows some motivating examples. Section 6 presents the object retrieval algorithms. In Section 7, we present our experimental results. Finally, Section 8 summarizes the conclusions of our work

## 2. Related Work

Organization of data on broadcast channels has been discussed in [5, 6, 7]. Chehadeh and Hurson [6] proposed clustering of related data objects along parallel channels.

Page retransmission and organization of replicated objects on broadcast channels has been explored. Wong discusses retransmission of pages over single channel broadcast cycles [10]. It was shown that, if  $L$  is the length of the broadcast, and  $q_1 \geq q_2 \geq \dots \geq q_n$  are the probabilities with which pages 1 to  $n$  (objects 1 to  $n$ ) are accessed, then the number of times,  $k_i$ , each page is transmitted should follow the following constraints:

- $\sum k_i = L$ , if  $k_i$  is the number of copies of  $p_i$
- $k_i/k_j$  should be as close to  $\sqrt{q_i}/\sqrt{q_j}$ , and
- replicas should be spaced as close to  $L/k_i$  as possible on the broadcast channel.

Hsu et al. [9] extends Wong's proposal for cyclic retransmission to multi-channel broadcasts. In their replication scheme, chunks of the broadcast, called Multiple Broadcast Segments (MBS), are replicated. In both the Hsu and Wong schemes, replicas are equally spaced to improve access time.

Sun et al. [1] proposed the Parallel Object Scan (POS) and Serial Empty Scan (SES) algorithms that produce optimal access patterns for retrieving objects efficiently from parallel air channels. However, these algorithms did not consider object replication on broadcast cycles. In our work we propose algorithms that can optimize object retrieval in the presence of replicated objects.

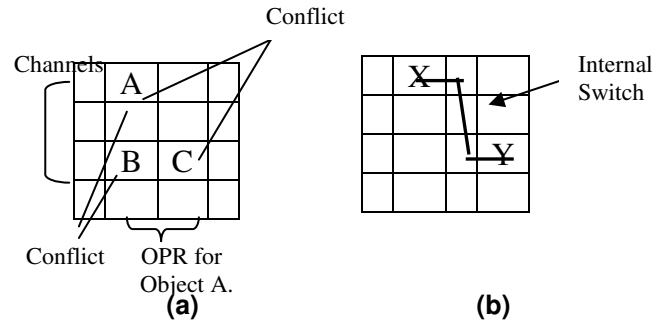
### 3. Preliminaries

A broadcast consists of a number of parallel channels. Each channel has a number of pages, or time slots in which a data object appears on that channel and can be read from. If the total number of parallel channels is  $NUM\_CHANNELS$  and the total number of pages in each channel is  $NUM\_PAGES$ , then the broadcast can be represented as a two dimensional  $NUM\_CHANNELS$  by  $NUM\_PAGES$  matrix (Figure 1(a)). In our work, each object spans only one page and is not fragmented across pages. Larger objects can be split across multiple pages and each such sub-object can be treated as different objects all of which must be retrieved.

A mobile unit can tune into only one air channel at a time. Thus, a mobile unit that attempts to read data from multiple channels must switch between these channels during a broadcast cycle (Figure 1(b)) generating an *internal channel switch*. Channel switching does not occur instantaneously. Without loss of generality, we assume that the mobile unit requires one page unit of time to switch channels. While a mobile unit is switching channels, it cannot read any data from any channel. Thus, if a mobile unit is reading an object  $o$ , it cannot read another data object in  $o$ 's *Overlap Page Region (OPR)* [1]. The value of OPR in our work is 2, i.e., for an object in page  $i$  on channel  $c$ , its OPR includes pages  $i$  and  $i+1$  on all channels other than channel  $c$  (Figure 1(a)). When

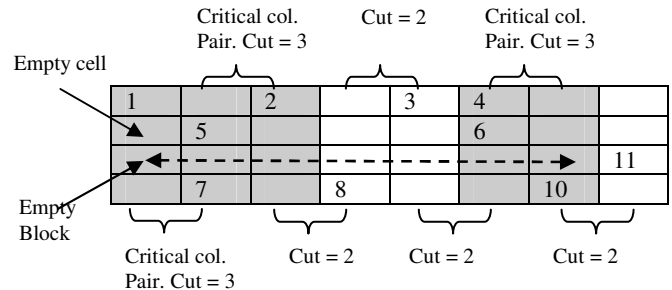
two objects are present in the OPR of each other, they are said to be in *conflict*.

Due to conflicts, a mobile client cannot retrieve all desired objects during a single broadcast cycle (Figure 1(a)). As a result, multiple *passes* over the broadcast are required. Whenever the mobile unit starts a pass over the broadcast cycle from a channel different from the channel where the previous pass ended, an *external switch* is generated.



**Figure 1. (a) Overlap Page Region (OPR) and Conflicts between objects. (b) Internal channel switching. The line represents a 'pass' made by the mobile unit through the broadcast for retrieving the requested objects.**

In the broadcast matrices shown in the rest of the paper, we only show objects that the mobile unit has to retrieve. **The minimum number of passes required to retrieve all the desired objects from the broadcast is determined by the max-cut value** [1]. If  $S$  is the set of objects being requested, the *cut at column  $j$*  is defined as  $|K_j|$ , where  $K_j = \{i \mid \exists j', j \leq j' \leq j+OPR-1, \text{ such that cell } C_{ij'} \in S\}$  [1]. Max-cut is defined to be *the maximum among all the cuts for the broadcast containing only the desired objects* (see Figure 2). Since, OPR in our work is 2, we refer to cut in terms of *pairs of adjacent columns*. We refer to the pairs of columns where the cut is equal to max-cut as *critical column pairs*. A page that does not contain any *desired* object is regarded as an *empty cell*; a contiguous set of empty cells is called an *empty block*.



**Figure 2. Cut and Max-cut. Max-cut = 3.**

To minimize power consumption, use of indexes for air channels has been proposed [3, 4, 9, 13]. A broadcast index provides information about when a data object

would occur on the broadcast. The mobile unit can schedule object retrieval using the index and switch between different modes of power consumption such as *doze mode*, when the unit is inactive and is simply waiting for the data to be broadcast, and *active mode*, when the unit is actively retrieving data from the channel.

#### 4. Problem Formulation

An object in a broadcast is represented as a triplet  $\langle id_i, c_i, p_i \rangle$ , where  $id_i$  is the *unique identifier* of the object  $o_i$ ,  $c_i$  is the channel and  $p_i$  is the page on which  $o_i$  is broadcast. Let  $S$  be the set of requested objects in the broadcast cycle. Let there be a set of requested objects that are replicated,  $S_R$ . Hence,  $S_{NR} = S - S_R$  is the set of all non-replicated objects in  $S$ . The problem at hand is to select a set  $S_R'$  such that:

- $S_R' = \{ \langle id_i, c_i, p_i \rangle \mid \langle id_i, c_i, p_i \rangle \in S_R \text{ and for all } i \neq j, id_i \neq id_j \}$ , i.e.,  $S_R'$  contains exactly one copy of every object in  $S_R$ .
- Retrieving  $S_{NR} \cup S_R'$  from the broadcast satisfies the following optimization criteria:
  - *Primary Optimization Criterion*: The number of passes and hence the response time is minimized
  - *Secondary Optimization Criterion*: For the given number of (minimal) passes, the number of internal switches, and hence the energy consumption, is minimized

#### 5. Motivating Examples

1						5	
	2		2	3			6
							7
	4		5				

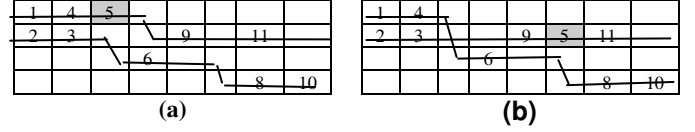
Removing these copies from the broadcast reduces max-cut from 3 to 2.

**Figure 3. Max cut reduction. Shaded columns are the critical columns.**

We show in Figure 3 how the choice of the replica retrieved influences the number of passes. The max-cut initially is 3. The replicated objects 2 and 5 occur in critical columns ( $\langle 2,2,2 \rangle$  and  $\langle 5,1,7 \rangle$ ), and in non-critical columns ( $\langle 5,4,4 \rangle$  and  $\langle 2,2,4 \rangle$ ). If the copies of 2 and 5 in the critical columns are chosen for retrieval, then the max-cut remains the same and three passes are required to retrieve all desired objects. However, if copies  $\langle 5,4,4 \rangle$  and  $\langle 2,2,4 \rangle$  are retrieved, then only two passes would be necessary. Thus, the challenge is to identify and remove those copies that affect the max-cut, and hence the total number of passes required.

Now, consider an example in which an object, say 5, has two copies  $\langle 5,1,3 \rangle$  and  $\langle 5,2,6 \rangle$  (Figures 4(a) and 4(b)). As can be noted the choice of replica generates different number of internal switches. Thus, the other

challenge is to identify those copies which when retrieved generate minimal internal switches.



**Figure 4. Effect of replicas on internal switches. (a) Retaining  $\langle 5,1,3 \rangle$  generates 3 switches. (b) Retaining  $\langle 5,2,6 \rangle$  generates 2 switches. Each line represents one pass, and everytime the unit changes over to another channel, an internal switch is generated.**

Our proposed algorithms only consider internal switches because typically the number of external switches is much fewer than the number of internal switches [1]. For optimizing the number of external switches, Sun et al. [1] have proposed a Minimal Spanning Tree (MST) approach.

#### 6. Proposed Object Retrieval Algorithms

We present two algorithms in this section. The greedy algorithms proposed in section 4.1 and 4.2 give sub-optimal solutions.

##### 6.1. The Greedy Max-Cut Algorithm

The Greedy Max-cut Algorithm (GMA) iteratively reduces max-cut (and consequently the number of passes) by removing certain replicas from critical columns.

The pseudo-code for the algorithm is shown below:

**Algorithm 1: Greedy Max-Cut Algorithm.**

// Greedy Max-Cut() reduces the max-cut by removing // replicas from Critical Columns.

**Input:**  $NUM\_CHANNEL$  by  $NUM\_PAGES$  matrix,  $BC$ .

**Output:**  $BC$  with reduced max-cut.

$CC \leftarrow \{c_b, \dots, c_k\}$  Get Critical-Columns ( $BC$ ).

$R \leftarrow \{R_b, \dots, R_k\}$ , where  $R_i = \{r_{i1}, \dots, r_{ik}\}$  is set of removable objects in critical column pair  $\{c_b, c_{i+1}\} \in CC$ .

$mc \leftarrow$  ,max-cut for  $BC$ .

do{

  /\* Reduce max-cut of  $BC$  by one\*/

**while** ( $CC$  is not empty **and** each critical column pair in  $CC$  has at least one removable object) {

    /\* Remove objects from those critical column pairs in  $CC$  that have only one removable object\*/

$SC \leftarrow$  Get all critical column pairs  $\{sc_b, sc_{i+1}\} \in CC$  where  $|R_i| = 1$ .

**if** ( $SC$  is not empty):

**for** (each critical column pair  $\{sc_b, sc_{i+1}\} \in SC$ )

$R \leftarrow R - R_i$

$SC \leftarrow SC - \{sc_b, sc_{i+1}\}$



5(b). Removable objects in critical column pairs are identified (marked by “\*” in the figures).

		8*			6*		1			
6	12	9			7	5	4			
		10*			2					11
8									10	9

5(c). Removing replicas that are the only removable objects in a critical column pair.

		8*					1			
6	12	9			7	5	4			
		10*			2					11
8									10	9

5(d). Iterative removal of single choices.

							1			
6	12	9			7	5	4			
		10*			2					11
8									10	9

5(e). Applying the least cut rule. New set of critical columns is computed.

Figure 5. Running example to illustrate GMA.

As a further illustration of the least-cut rule, consider the example in Figure 6(a). Objects 8, 9, 10 are removable objects in critical columns. In iteration 1, one of the four objects in the critical column pair is to be removed. The least cut at which object 8 has a copy is 3, least cut at which object 9 has a copy is 2 and the least cut at which object 10 has a copy is 1. So, applying the least cut rule, object 10 is removed and the max cut comes down to three. On repeating this process, the max-cut can finally be reduced to two. Figure 6(d) demonstrates that if the least-cut rule wasn't applied, the max-cut could not have been reduced beyond 3.

		8*								
		9*		9	7		8			
		10*			2		12			
		11					13		10	

6(a). Example Broadcast. Max-cut is 4.

		8*								
		9*		9	7		8*			
					2		12			
		11					13		10	

6(b) Remove object 10 from critical column pair according to least cut rule and recomputed new critical columns. Max-cut becomes 3.

		8								
				9	7					
					2		12			
		11					13		10	

6(c) Reapply least-cut rule. Remove objects '9' and '8' from critical column pairs and max-cut becomes 2.

		9*		9	7		8			
		10*			2		12			
		11					13		10	

6(d) If least-cut rule is not followed, Max-cut cannot be reduced beyond 3.

Figure 6. Illustration of significance of the Least-Cut Rule.

### 6.2. The Greedy Block Algorithm for reducing the number of internal switches

The Greedy Block Algorithm (GBA) aims at reducing the number of internal switches. After application of the GMA, we are left with a set of replicated objects  $S_{GMA}$  such that  $S_{GMA} \subseteq S_R$ . After running the GBA, we have the set of replicated objects,  $S_R'$  (Section 3.2).

The heuristic used by the GBA for selecting one copy from many copies of a replicated object follows the largest empty block rule:

**Rule 1: Largest Empty Block Rule:** At every step, remove those copies of replicas that result in the formation of the largest empty block in the broadcast matrix.

Consider a segment of a channel  $c_i$  where  $k$  replicated objects,  $o_1$  to  $o_k$ , occur between page  $p_1$  to page  $p_n$  without any non-replicated objects in between (empty cells might occur in between). The largest empty block that can possibly be formed when objects  $o_1$  to  $o_k$  are removed consists of pages  $p_1$  to  $p_n$  and all empty cells immediately preceding page  $p_1$  (if any) and all empty cells immediately after  $p_n$  (if any). Since these objects are all replicated and have other copies in the broadcast, such a block can potentially be “emptied” — called a *potential block* (See figure 7(a)).

The intuitive idea behind the largest empty block rule is that, by removing objects that result in the formation of large empty blocks, we are trying to select those replicas that are closer to other objects on a channel. This increases the chances that these copies can be read in the same broadcast cycle without additional switches. An object flanked by large empty blocks, increases the chances that the mobile unit switches to the channel in which the object is present, and then switches again to another channel to read objects, resulting in more internal switching. If a replicated object is flanked by large blocks, removing it could reduce the number of switches.

In every step, the Greedy Block Algorithm computes all possible ‘potential blocks’ and removes the largest among them (See Figure 7).

		1				2			3	13	
4	5		1			2		6			7
				9	10			6		11	6
	15		13				14				

(a). Potential blocks (shaded blocks).

		1				2			3	13	
4	5										7
				9	10			6		11	6
	15		13				14				

(b). Applying Heuristic1, potential block on channel 2, page 3 to 11 is removed. Potential blocks are recomputed in every iteration.

		1				2			3	13	
4	5										7
				9	10			6		11	6
	15						14				

(c). Apply largest empty block rule iteratively.

		1				2			3	13	
4	5										7
				9	10					11	6
	15						14				

(d). Final Result with  $S_R'$ .

Figure 7. Application of Heuristic1.

If two potential blocks are of the same size, then the rightmost block rule is followed.

**Rule 2. Rightmost block rule:** *If two (or more) potential blocks have the same size, then the “rightmost” potential block that starts later in the broadcast cycle, is chosen for removal.*

The rightmost block rule tries to remove the blocks that occur later on in the broadcast. By retaining replicas that occur earlier, the rightmost block rule tries to end the retrieval process earlier, that is, the last object retrieved by the client occurs earlier in the broadcast, thereby improving response time. Figure 8 illustrates the rightmost block rule.

1	2		3	4							
							5	2		3	

Figure 8. Rightmost block rule. If the rightmost block is removed, then retrieval ends at page 8, otherwise, retrieval ends at page 11.

The running time for this algorithm is  $O(C * P * N)$  where  $N$  is the total number of replicated objects in the broadcast. Again, the worst-case running time for this algorithm is  $O((C * P)^2)$  as there can be at most  $C * P$  replicated objects.

---

### Algorithm 2. Greedy Block Algorithm.

// Greedy Block Algorithm tries to minimize the number // of switches by applying the Largest Empty Block Rule.

**Input:**  $NUM\_CHANNEL$  by  $NUM\_PAGES$  matrix  $BC$ .  
List of replicated Objects  $ROL$ .

**Output:**  $BC$  containing only one copy of every requested object ( $S_R'$ ).

**Until** only one copy of every object remains in  $ROL$  **do**  
**for**  $i=1$  to  $NUM\_CHANNELS$

$PB[i] \leftarrow$  Get largest Potential Block in channel  $i$   
(apply the Rightmost Block Rule if necessary).

/\* Remove largest block in the entire matrix  $BC$ \*/

$LB \leftarrow$  Get largest block in  $PB$  (apply the Rightmost Block Rule if necessary).

Remove objects in  $LB$  from  $BC$ .

Remove from  $ROL$  replicas in  $LB$ , and objects that no longer have multiple copies.

---

### 6.3. Branch and Bound

We implemented a branch and bound algorithm to obtain the solution that gives the optimal number of passes, and for those passes, the minimum number of switches. For running the branch and bound algorithm, the broadcast matrix is initialized with only non-replicated objects first. Then at every step, the algorithm adds *one copy of one requested replicated object* and runs the POS algorithm to obtain the new number of switches and passes. The number of passes and switches obtained are checked against the bounds on passes and switches to prune out worse solutions. The bound can be initialized to arbitrarily large values or the sub-optimal values obtained by the greedy algorithm to prune out a large part of the search space.

To make the branch and bound algorithm faster, we exploit the monotonicity property described in Theorem 1. If the number of passes is same as the bound on passes and the number of switches is same as the bound on the number of switches, the configuration is still ignored (step 14 in the branch and bound algorithm), that is we ignore solutions that result by adding the remaining copies of replicated objects. This is because addition of more objects will not yield a better solution than what we already have (Theorem 1).

**Theorem 1. (Monotonicity Property)** *Given a broadcast matrix  $BC$  consisting of a set of desired objects  $S$  to be retrieved, if the number of switches required to retrieve all objects in  $S$  is  $n$ , then the addition of a new object  $o$  to  $S$  that does not increase the max-cut of  $BC$ , will result in a number of switches  $n'$  such that  $n' \geq n$ .*

**Proof:**

Let  $n$  be the least number of internal switches required for retrieving all objects in  $S$  from  $BC$ . Consider a new

object  $o$  that is added to  $S$  which does not increase the max-cut. Let  $n'$  be the least number of internal switches required for retrieving objects  $S \cup \{o\}$ .

Assume that  $n' < n$ . Now the access schedule for retrieving  $S \cup \{o\}$  is a valid access schedule for retrieving  $S$  as well, since addition of a new object  $o$  can only increase, but not reduce conflicts in BC. So the access schedule for retrieving  $S \cup \{o\}$  is also a valid schedule for retrieving  $S$  and so  $n'$  and not  $n$  would not be the least number of switches for retrieving  $S$ . This is a contradiction. So addition of new objects can never reduce the number of internal switches required.

---

**Algorithm. Branch-And-Bound.**

// Branch-and-Bound() finds the optimal solution for  
//object retrieval with minimum passes, and for those  
//passes, minimum switches.

**Input:**  $NUM\_CHANNEL$  by  $NUM\_PAGES$  broadcast matrix  $BC$ .

**Output:** Optimal number of passes and internal switches as per optimization criterion described in Section 3.2.

```

1. BoundPasses ← Initialize.
2. BoundSwitches ← Initialize.
3. BC ← Broadcast with all replicated objects removed.
4. i ← Initialize to 0.
5. L ← List of replicated objects 1 to k and their various replicas in the broadcast.
6. Until all nodes in L[1] have been visited{
7. Insert replica in node n of L[i] that has not been visited earlier into BC.
8. Mark this node of L[i] as 'visited'.
9. If (Max-cut of BC > BoundPasses){
10.   Remove n from BC.
11.   Continue.
}
12. else{
13.   Switches ← Run POS Algorithm to obtain number of switches.
14.   If (Max-cut of BC < BoundPasses OR (Max-cut of BC == BoundPasses AND Switches < BoundSwitches){
15.     i ← i + 1
16.     if (i == k){
17.       BoundPasses = Max-cut of BC.
18.       BoundSwitches = Switches.
} else{
19.   Remove n from BC.
}
}
}

```

The average running time for the branch and bound algorithm is  $O(R^N * N * C^2 * P)$ , where N is the number of

replicated objects, R is the maximum number of replicas in the matrix, C is the number of channels, P is the number of pages in each channel and  $C^2 * P$  is the complexity of the POS algorithm. As the number of replicas increase, the running time of the algorithm increases exponentially.

## 6.4 Other approaches for object retrieval

Several simple heuristic algorithms can be used to schedule objects retrieval in the presence of object replicas. Among the replicas of an object, the *Select-First* algorithm, always chooses to retrieve the first copy of a replica as pointed by the index. The *Select-Random* algorithm randomly selects a replica for retrieval among the several replicas available.

Both the *Select-First* and *Select-Random* approaches have a running time of  $O(N)$ , where N is the number of objects the mobile unit retrieves.

## 7. Experiments and Results

In this section, we present the results of our empirical evaluation of the different algorithms proposed above.

### 7.1. Experimental/Simulation Set Up

We chose three different schemes for organizing replicated objects on the broadcast. In real life, it is likely that data is organized on broadcast channels in a way that minimizes response time [9, 10]. The three schemes (Section 2) we used are (Table 2):

- Wong scheme [10] for page retransmission.
- Hsu scheme [9] for object replication.
- A random distribution of replicas on the broadcast.

The Wong and Hsu schemes use object access frequencies for organizing replicas. To make our model more realistic, the access frequencies of the objects were obtained using the Zipf distribution [10, 11]. There is empirical evidence that the way data is accessed in broadcast environments such as the Teletext System [11] follows the Zipf distribution.

**Table 1. Input Parameters to the Simulator.**

Parameters	Values	Parameters	Values
Broadcast Length	4290 Pages	Transmission Rate	131072 Bytes/sec
Page Size	512 Bytes	Index Type	Inheritance Index
Index Size	297015 Bytes	Index Size	297015 Bytes
Doze Mode Power	6.6 mJ	Zipf Parameter	1.0
Active Mode Power	130 mJ	No. of simulation runs	25
Switching Mode Power	13 mJ		

We assume that all the parallel channels have the same broadcast cycle length. Access frequencies are used for organizing data on the channels. The client randomly chooses objects for retrieval in each simulation run.

In our simulation, we include all events on the client side: **(a)** Wait time for an index to appear on a channel. **(b)** Time/Energy to read the index. **(c)** Time/Energy spent waiting for objects to be broadcast. **(d)** Time/Energy for retrieving objects from the broadcast.

The simulation data and parameters shown in Table 1 are similar to those reported by Sun, et al. [1].

**Table 2. Experimental set up.**

Experimental Set-up	Replication Scheme	No. of objects broadcast	No. of channels
1	Wong	420	10
2	Hsu	60	2-15
3	Random	4290	10

## 7.2. Experimental Results

### 7.2.1 Number of passes for varying input parameters.

Figure 9(a) shows the number of passes for the proposed algorithms for Experimental set-up 1. As can be seen, Greedy algorithm produces results close to optimal solution, while Select-First and Select-Random require far more passes. In addition, as expected, the number of passes increases as the number of requested objects increases. This is justifiable because increasing the number of the requested objects increases the number of conflicts, hence increasing the number of passes. Similar results for a random distribution of replicas on the broadcast were observed (Figure 9(c)).

Figure 9(b) shows the effect of varying the number of parallel air channels on the number of passes required for retrieving 10 objects. As can be seen, as the number of channels ( $N$ ) increases, the number of passes required also increases. This is because, as  $N$  increases, for a given broadcast length (4290), the length of each channel ( $4290/N$ ) decreases; as a result, the number of conflict increases requiring more passes to resolve a request. In this experiment also Greedy Algorithm performs very similar to optimal solution.

**7.2.2. Response time.** Figure 11 shows the response times for the various algorithms when retrieving a varying number of objects for Experimental set-up 1. As can be seen, Optimal and Greedy algorithms have the lowest response time as compared to Select-First and Select-Random heuristics. This is in tune with our earlier results (Figure 9(a)) — as the number of passes increase, the response time also increases.

**7.2.3. Internal Switching.** The performance of our algorithms in terms of the number of internal switches is

shown in Figure 13. Greedy algorithm requires the highest number of internal switches. In comparison with the Select-First and Select-Random heuristics the Greedy algorithms uses fewer passes, and fewer passes over the broadcast requires more channel switches in order to accesses as many objects as possible in each pass. Optimal algorithm also requires a larger number of internal switches for the same reason.

**7.2.4. Energy consumption.** Energy consumption of the mobile unit primarily consists of energy consumed during active and doze operational modes, and the energy consumed for channel switching. Since the number of objects being retrieved at any time is the same for all algorithms, the energy consumption during the active mode is the same for all of them. Energy consumption due to the doze mode depends heavily on response time and the number of passes, while switching energy would vary as per the number of internal switches.

Figure 10 shows the energy consumption when number of requested objects varies. Figure 10(a) illustrates the energy consumed by internal switching. This result is in tune with the results in Figure 13.

Figure 10(b) demonstrates the energy consumption during the doze operational modes. Due to fewer number of passes and lower response time, greedy and optimal algorithms incur the least doze mode energy consumption. In figure 10(c), we show the combined energy consumption for internal switching and doze mode. The energy saved by Greedy algorithm in doze mode compensates for much of the extra energy consumed during internal switching. As can be seen from the figure, greedy algorithm does fairly well in terms of net energy consumption when compared with the heuristic algorithms.

**7.2.5. Run time.** From the complexity analysis for branch and bound algorithm, it is clear that run time increases exponentially as the number of objects requested increase. Greedy Algorithm has a significantly much lower running time than branch and bound. A comparison of the running times is shown in Figure 12.

## 8. Conclusions

Four approaches for efficient object retrieval from parallel broadcast channels in the presence of replicated objects were proposed and their performance was evaluated. The Select-First and Select-Random algorithms gave poor performance in terms of number of passes, response time, and doze-mode energy consumption. The Greedy algorithm resulted in near optimal results in terms of number of passes, response time, and doze mode energy consumption. Although the greedy algorithm resulted in

higher internal switching energy consumption as compared to the other three approaches, that was compensated by the reduced doze-mode energy consumption. Furthermore, the greedy algorithm has a much lower running time (3 orders of magnitude) when

compared to branch and bound algorithm that gives optimal solutions.

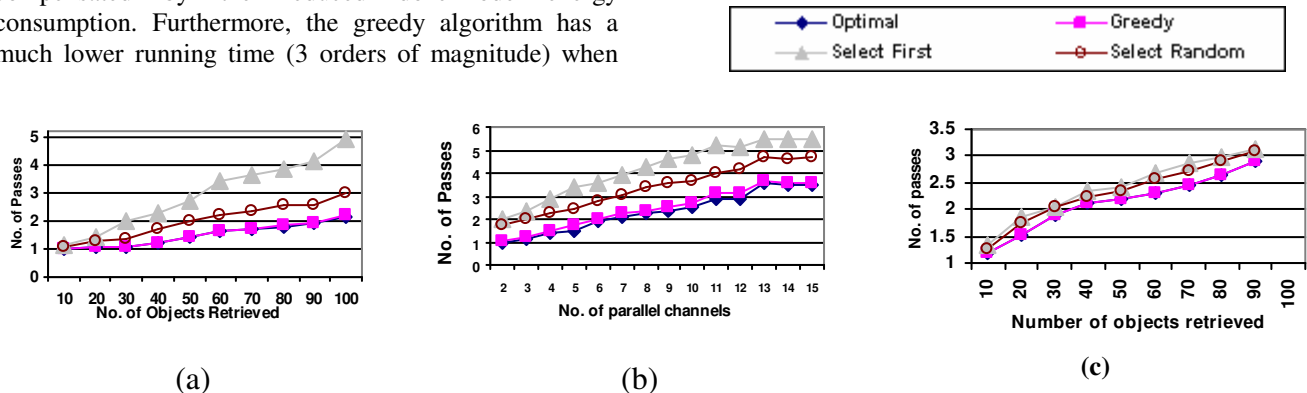


Figure 9. (a) No. of passes vs. No. of requested objects (Set-up 1). (b). No. of passes vs. No. of parallel channels (Set-up 2). (c) No. of passes vs. No. of requested objects (Set-up 3).

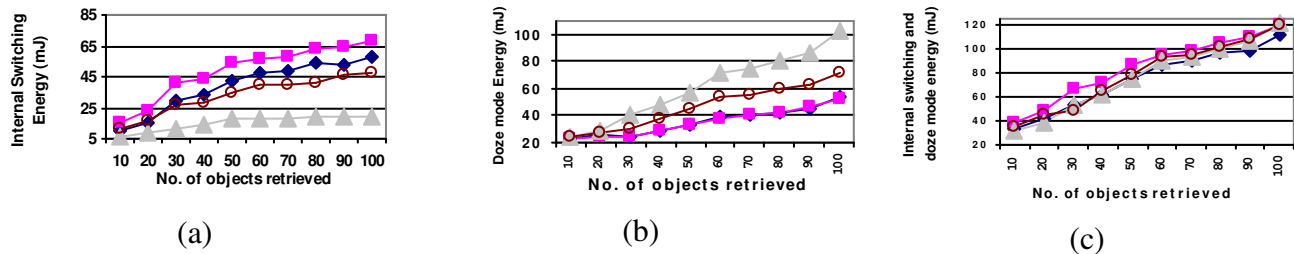


Figure 10. Energy consumption vs. No. of requested objects. (a). Internal Switching energy (Set-up 1). (b). Doze mode energy (Set-up 1). (c). Doze mode and internal switching energy combined (Set-up 2).

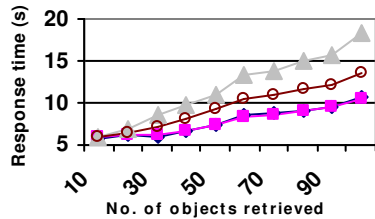


Figure 11. Response times vs. No. of objects (Set-up 1).

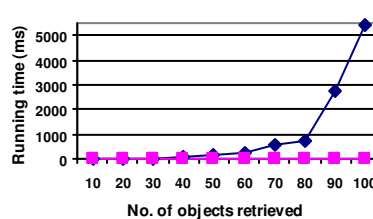


Figure 12. Execution time (Set-up 1).

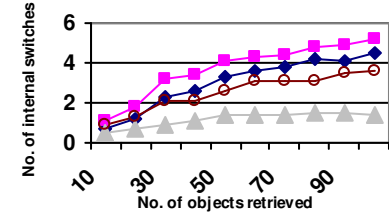


Figure 13. No. of internal switches vs. No. of requested objects (Set-up 1).

## 9. References

- [1] Sun, B., Hurson, A., R., & Hannan, J. "Energy efficient algorithms for object retrieval on parallel broadcast channels". In Int. Conf. on Parallel Proc. (ICPP), 2004.
- [2] Triantafillou, P., Harpantidou, R., & Paterakis, M. "High Performance Data Broadcasting: A Comprehensive Systems' Perspective." In MDM, 2001.
- [3] Hu, Q., Lee, W., & Lee, L., D. "Indexing techniques for wireless data broadcast under data clustering and scheduling". In CIKM 1999.
- [4] Tomasz, I., Viswanathan, S., & Badrinath, R., B. "Energy Efficient Indexing on Air". In ACM SIGMOD International Conference on Management of Data, 23(2):25-36, 1994.
- [5] Chehadeh, C.Y., Hurson, A.R., and Kavehrad, M., "Object Organization on A Single Broadcast Channel in the Mobile Computing Environment," Multimedia Tools and Applications Journal, 9:69-94, 1999.
- [6] Chehadeh Y.C. and Hurson, A.R. "Object Organization on Broadcast Channels in a Global Information Sharing Environment". Journal of Information Science and Technology, 2000.
- [7] Juran, J., Hurson, R., A., Vijayakrishnan, N., & Kim, S. "Data Organization and Retrieval on Parallel Air

- Channels: Performance and Energy Issues.” *Wireless Networks*, 10:183-195, 2004.
- [8] Hameed, S., & Vaidya, H., N. “Efficient Algorithms for Scheduling Data Broadcast.” *Wireless Networks*, 5:183-193, 1999.
- [9] Hsu, C., Lee, G., & Chen, A., L. “Index and Data Allocation on Multiple Broadcast Channels Considering Data Access Frequencies.” In the Third International Conf. on Mobile Data Management (MDM), 2002.
- [10] Wong, J. (1998). “Broadcast Delivery.” In *IEEE*, 76(12):1566-1577.
- [11] Ammar, M., & Wong, J. On the Optimality of Cyclic Transmission in Teletext Systems. In *IEEE Transactions on Communications*, 1987, 35(1):68-73.
- [12] Konstantinos, S., Roussopoulos, N., & Baras, J., S. “Adaptive Data Broadcast in Hybrid Networks”. In 23<sup>rd</sup> VLDB Conference, Athens, 1997.
- [13] Jung, S., Lee, B. & Pramanik, S. “A Tree-Structured Index Allocation Method with Replication over Multiple Broadcast Channels in Wireless Environments.”, In *IEEE Trans. Know. and Data Eng.*, 17 (3), pp 311 – 325, 2005.
- [14] Huang, Y., Sistla, P., & Wolfson, O. “Data Replication for Mobile Computers”, *SIGMOD*, Minneapolis, USA, 1994.