Journal of Sustainable Computing, to appear, 2016

Energy-Efficient Algorithms for Data Retrieval from Indexed Parallel Broadcast Channels

Ali R. HURSON, Sahra SEDIGH, and Mike WISELY

***Abstract*--Constraints on the energy, bandwidth, and connectivity of mobile devices and wireless communication medium complicate the timely and reliable access to public data. Energy is often the most stringent constraint, necessitating techniques that facilitate operation in energy-saving modes. Broadcasting, typically over parallel channels, has proven to be an effective method for dissemination of public data to mobile devices. However, the employment of parallel channels introduces challenges associated with channel switching and conflicts due to concurrent accesses to multiple data items that ultimately increase energy consumption and response time. The detrimental effects on energy consumption and response time can be alleviated by scheduling the retrieval of data items in an order that reduces the number of passes over the air channels and channel switching between the parallel channels.**

In this paper, several scheduling algorithms are proposed and analyzed that achieve the aforementioned objectives. To further improve energy consumption and response time, the scope of our scheduling algorithms has been enhanced by replication of popular data items. The proposed scheduling algorithms, both with and without replication, have been simulated, and simulation results are presented and analyzed. These results show that the proposed scheduling algorithms, compared to some heuristic based methods, have greater impact in reducing energy consumption and response time. This reduction is shown to be more pronounced with replication of data items.

***Index Terms*: Algorithm/protocol design and analysis, query processing, retrieval models, ubiquitous computing, energy management, broadcasting**

I. INTRODUCTION

THE advent of *mobility*, in which a user accesses data and services through a wireless medium with limited resources, has introduced additional complexity to traditional approaches in database systems. In current practice, regardless of the underlying computational model, hardware devices, and connection media, data sources can be classified as follows:

- Private data, e.g., personal daily schedules and phone numbers. The reader of this type of data is mainly the sole owner/user of the data.
- Public data, e.g., news, weather information, traffic information, and flight information. This type of data is maintained by one source, and shared by many. Consequently, a user mainly

Ali. R. Hurson and Mike Wisely are with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409 (telephone: 573-341-6201, e-mail:{hurson,mwwcp2}@mst.edu).
S. Sedigh is with the Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65409, e-mail sedighs@mst.edu.

queries the information source(s).

- Shared data, e.g., traditional replicated and/or fragmented databases. A processing node may actually contribute to maintaining consistency of and participate in distributed decision making with this type of data ─ a user usually sends transactions as well as queries to the information source(s).

The increasing ubiquity of mobile devices, such as smart phones, and proliferation of global positioning systems have brought the vision of ubiquitous and anytime, anywhere access to public data closer to reality. Broadcasting has been proposed as an efficient method to disseminate data to a large number of users [1]. Broadcasting is invaluable during and after catastrophic events, as it can guide evacuation, search, and rescue efforts. Broadcasting can be used to disseminate location information and schedule updates in venues such as airports, large shopping malls, or amusement parks. Live video streaming that has gained popularity can take advantage of data broadcasting. Travel and Traffic Information (TTI) systems, whether based on decentralized inter-vehicle communication over radio [19], or centralized digital radio systems [9], also can take advantage of broadcasting. It is anticipated that future evolutions of ground transportation system will ultimately end up being a Cyber-Physical System (CPS). In such an environment, broadcasting of traffic information and traffic pattern to either manned or unmanned vehicles can play a significant role to increase safety and reduce travel cost.

Services similar to Microsoft Network (MSN) Direct which used FM radio to deliver location-based services to navigation systems are other prime examples of applications where broadcasting would be the most efficient means of dissemination of data. The information available through MSN Direct in large urban areas in North America included traffic reports, gas prices, weather reports, stock quotes, and event schedules [17].

Hybrid communication platforms that use both the cellular infrastructure and wireless radio facilitate a wide range of broadcasting-based services [9]. The extension of MSN Direct to the Windows Mobile Platform is an example. The same concept could be applied to wireless sensor networks, both in fixed installations and in rapid deployments following events such as wildfires.

The aforementioned applications are dissemination–oriented, and their data access and data flow characteristics differ considerably from those of traditional client-server database applications. Asymmetry of communication (as the majority of the traffic is retrieval of data items by the mobile devices (downlink), and the uplink carries minimal (or sometimes no) data [1]) and significant overlap of requests (e.g., live video stream) are distinct characteristics of broadcast based applications. Mobile data devices such as portable navigation systems or other handheld devices with wireless communication capability are typically switched on by the user sporadically, as needed for retrieval of information, and then powered off, precluding the use of intelligent caching strategies for efficient data retrieval [2].

Advances in the development of temporary energy sources have not kept pace with the increasing ubiquity of handheld mobile devices, and as such, the energy efficiency of data retrieval by mobile devices is likely to remain a concern for the foreseeable future. To manage energy consumption, one needs to develop energy-efficient protocols that run hardware devices in different operational modes [8, 10-12]. The literature has suggested indexing techniques and broadcasting over parallel channels as means to reduce the active time of mobile devices and the broadcast length ─ hence reducing the *energy consumption* and the *response time*. However, *access conflicts* and *channel switches* are natural by-products of broadcasting over parallel channels [10]. Access conflicts (see Section II.D) force additional passes over the parallel channels, which in turn increase the response time and energy consumption. Moreover, channel switching incurs additional energy consumption at the mobile device. As a result, in the presence of conflicts, one must develop energy efficient protocols for prudent scheduling of accesses to the requested data items in an attempt to reduce the number of broadcast passes and channel switches.

Within the scope of indexed parallel broadcast channels, heuristic rules were employed to schedule access to the data items such that both the number of passes and the frequency of channel switching were reduced [10]. Simulation results have proven the effectiveness of these approaches. However, as expected, these heuristic-based algorithms do not necessarily generate reduced response time and energy consumption in all cases.

Replication can be employed to increase the availability of popular data items on parallel broadcast channels. This technique has both advantages and disadvantages; replication should offer reduced response time and energy consumption if the replicated data items are requested, however, it increases the broadcast length, and hence the response time and energy consumption, for non-replicated data items.

This work extends the scope of our earlier investigation [10, 12]. It proposes and evaluates several scheduling algorithms that generate access patterns that reduce response time and energy consumption of data retrieval from indexed parallel broadcast channels. The proposed scheduling algorithms have been further simulated and studied for their energy consumption and execution time. Finally, data replication is employed, and its effect on energy consumption and response time is studied.

This article is organized as follows. Section II provides background information, discusses related studies, and formulates the problem. The proposed scheduling algorithms for data retrieval are articulated in Section III and Section IV reports on our efforts to employ replication.

The simulation results of the proposed algorithms are presented and analyzed in Section V. In Section VI, the conclusions are drawn, and future research directions are addressed. Finally, for the sake of completion and in light of space limitation, running examples of the proposed algorithms have been presented in the Appendix.

II. BACKGROUND AND RELATED WORK

A. *Wireless Computing*

The wireless computing architecture consists of both mobile and fixed devices. The network servers communicate with mobile devices through mobile support stations (MSSs), as depicted in Fig. 1. In this environment, one could recognize three types of services:

- Interactive/on-demand service: In this case, clients obtain answers to requests through a dialogue (two-way communication) with the database server(s): the user request on the uplink channel is pushed to the server(s), data sources are accessed, query operations are performed, partial results are collected and integrated, and the generated information is communicated back to the user ─ the channels are bi-directional and asymmetric. Issues related to on-demand services are beyond the scope of this work; the interested reader is referred to [5].
- Broadcast-based service: Many applications are directed towards public information characterized by (i) the multitude of users and (ii) the similarity and simplicity of the requests made. In such applications, information is generated and periodically provided to all users on the wireless channels. Mobile users are capable of searching the wireless channels and pulling their required data ─ the channels are unidirectional from servers to mobile devices. In the architecture of Fig. 1, each MSS can broadcast to mobile devices within its communication range. This broadcast could take place over the cellular phone infrastructure,

or through wireless radio.

- Pervasive service: In on-demand and broadcast-based services, the user initiates the process. In pervasive service, computers work in the background, intelligently and autonomously gathering the potential information and services, and making them available to the user.

B. Data Broadcasting

The main advantage of broadcasting is the fact that it scales up as the number of users increases, eliminating the need to multiplex bandwidth among users accessing the wireless channel. Furthermore, broadcasting can be considered as additional storage available over the air for the mobile devices. This is an attractive solution, due to the limited storage capability of a typical mobile device. Data on a broadcast channel is read-only and must be accessed sequentially. Data can be broadcast either on a single channel or on parallel channels. This work assumes that all parallel channels have the same transfer rate, and that data broadcasting will be carried out in a cyclic manner, i.e., data is re-transmitted at the end of each broadcast cycle with any updates that occurred during the previous broadcast cycle.

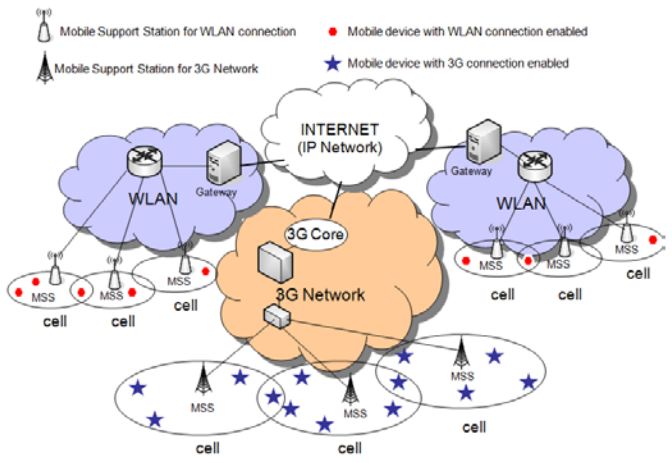

Mobile Support Station for WLAN connection
Mobile device with WLAN connection enabled
Mobile Support Station for 3G Network
Mobile device with 3G connection enabled
INTERNET
(IP Network)
WLAN
Gateway
3G Core
3G Network
MSS
cell

Fig. 1: Architecture of mobile computing environment.

Within the scope of broadcasting, research has been directed towards three different areas:

- how to determine the contents of the broadcast,
- how to reduce energy consumption at the mobile device, and
- how to reduce the response time.

Three models, namely, pull-based, push-based, and hybrid models, have been proposed to generate and disseminate the data items of interest to the users. Such an issue is beyond the scope of this work, and the interested reader is referred to [10] for further discussion. Application of various indexing techniques and employment of parallel broadcast channels are potential solutions to reduce response time and energy consumption. However, the employment of parallel channels introduces conflicts that necessitate multiple passes over the parallel wireless channels, increasing the energy consumption and response time. Therefore, it would be desirable to schedule the data retrieval during each broadcast cycle with the goal of reducing the number of passes. This issue is the major theme of this paper.

C. *Indexing Techniques on Wireless Channels*

An index is the auxiliary information pointing to the location or possible availability of a data item on the broadcast. With indexing techniques, the mobile device can predict the arrival time of requested data items, which facilitates prudent usage of energy-saving modes. The advantages of indexing schemes come at the expense of computational overhead and increased broadcast length, which could increase the response time and energy consumption. Within the scope of broadcasting, several indexing techniques, e.g., signature and tree-based schemes, have been proposed and studied in the literature [8].

In tree-based schemes, a metadata is generated that associates the contents of the data items with their respective locations. A single path throughout the tree shows the availability and location of the information requested. In contrast, signature-based indexing makes use of a signature, i.e., an abstraction of the information contained in a data item, to locate the information requested. Tree-based schemes have proven to be more suitable for reducing the overall energy consumption, as the index provides global information regarding the location of the data items on the channel(s) [12]. Based on this observation, this work employs tree-based indexing.

Several alternative layouts can be used to organize and index data on parallel broadcast channels. Refer to Fig. 2, where each row represents a parallel broadcast channel: the index can be broadcast once at the beginning of each broadcast cycle (Fig. 2(a)); it can be replicated and continually broadcast on a dedicated channel (Fig. 2(b)); or it can be replicated, interleaved with data items, and broadcast on a dedicated channel (Fig. 2(c)). Without loss of generality, this work will assume the layout of Fig. 2(a), due to its simplicity and clarity.

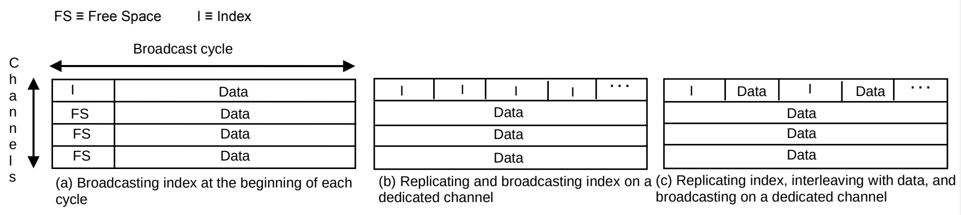

FS ≡ Free Space
I ≡ Index
Broadcast cycle
Channels
I
Data
FS
Data
FS
Data
FS
Data
(a) Broadcasting index at the beginning of each cycle
I
I
I
I
. . .
Data
Data
Data
(b) Replicating and broadcasting index on a dedicated channel
I
Data
I
Data
. . .
Data
Data
Data
(c) Replicating index, interleaving with data, and broadcasting on a dedicated channel

Fig. 2: Alternative layouts for tree-based indexing.

D. *Problem Formulation and General Protocol for Data Retrieval from Parallel Channels*

In this work, a matrix $\boldsymbol{C}$ of size $N \times M$ 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 $C_{i,j}$, $1 \leq i \leq N$, $1 \leq j \leq M$, represents a page, which is the smallest granule of data that can be retrieved from the broadcast channel. A row of cells represents a channel on the broadcast, while a column of cells represents pages transmitted concurrently on parallel channels. Without loss of generality, we assume that data items are not fragmented across adjacent pages, and that if one requests any data item in a page, the whole page will be retrieved.

The mobile device can tune in to only one channel at a given time. The mobile device can switch channels, but it takes time and consumes energy to do this. Based on the common page size and the network speed, the time required to switch from one channel to another during a broadcast cycle is not longer than the duration of one page in the broadcast. During this switching time no data can be read by the mobile device. We define the *overlapped page range* (*OPR)* as the minimum number of columns separating two requested items on different channels that can be read by the mobile device on the same pass. Based on the aforementioned discussion, we assume *OPR* = 1 for this work. We say that two data items are in *conflict* if it is impossible to retrieve both during the same broadcast cycle, necessitating two passes for their retrieval. Thus, given that *OPR* = 1, the data item $C_{i,j}$ is in conflict with $C_{k,j}$ and $C_{k,j+1}$, where $k \neq i$ (different row), $\forall\ 1 \le i, k \le N$ and $1 \le j \le M$-1. Fig. 3 shows the conflict area associated with a requested data item.

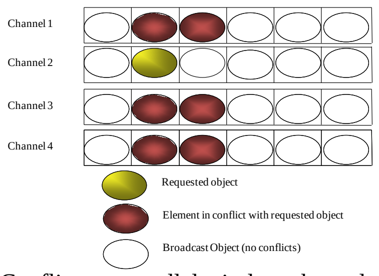

Channel 1
Channel 2
Channel 3
Channel 4
Requested object
Element in conflict with requested object
Broadcast Object (no conflicts)

Fig. 3: Conflicts on parallel wireless channels

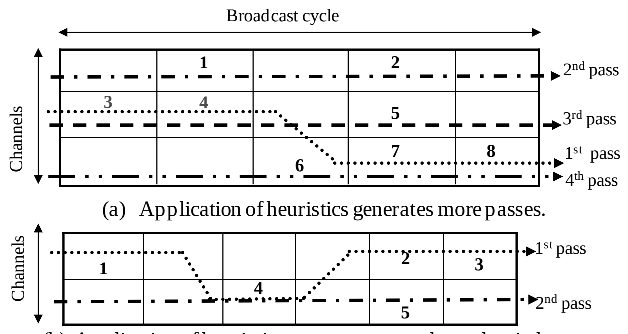
A general access protocol for data retrieval from parallel channels involves the following steps:

1. **Initial probe**: The client tunes into the channel hearing the index, in anticipation of the next broadcast of the index.
2. **Search**: The index is accessed to determine the offsets of requested data items.
3. **Compute**: The access patterns for the requested data items are generated.
4. **Retrieve**: The client, in active mode, tunes into the required channels (one channel at a time), downloads the required data items, and switches to sleep mode upon completion.

During the *compute* step, scheduling algorithms can attempt to reduce 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 switches are those occurring during a broadcast cycle. Outside switches occur between two successive broadcast cycles. In this work, we are interested in reducing the number of inside switches, since outside switches do not affect the number of passes.

Several heuristic-based algorithms have been proposed to schedule data retrieval during a broadcast cycle, including the solution to the Traveling Salesman Problem, and simple heuristics such as *Next Object* and *Row Scan* [12]. We reported on the employment of more sophisticated heuristic rules for generation of access patterns in [10]: first, temporal and spatial information present in the broadcast index was used to generate an “access forest” composed of “access trees,” where each tree represented an access pattern for the broadcast. A set of heuristics were subsequently employed to identify the trees resulting in the fewest passes and channel switches. It was found that these heuristic approaches, because of their very nature, were at times unable to generate “proper” access patterns that reduce energy consumption and response time for all circumstances.

Fig. 4 illustrates the shortcoming of these heuristic approaches. Assume a three-channel broadcast and a request for eight data items (Fig. 4(a)). Applying the heuristics reported in [10] results in four passes – in the 1st, 2nd, 3rd, and 4th passes, respectively, data items (3, 4, 7, 8), (1, 2), (5), and (6) are retrieved. However, a simple *Row Scan* approach would require only three passes (one over each channel) to retrieve the same set of data items. In Fig. 4(b), we assume a two-channel broadcast and a request for five data items. Employment of the heuristics of [10] requires three channel switches, as compared to the single switch required by a *Row Scan* approach.



(b) Application of heuristics generates more channel switches.

Fig. 4: Examples where the use of heuristics results in worse performance than row scan.

E. Replication of Data on Parallel Channels

Frequently accessed data items can be replicated and broadcast more than once during a broadcast cycle to increase their availability, and hence decrease the average response time. However, replication will increase the overall broadcast length, and as such, it is expected to increase the response time for non-replicated data items.

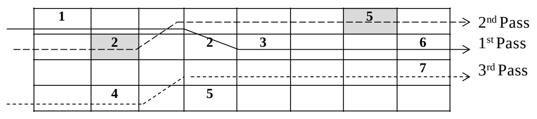
Several studies have addressed the transmission and organization of replicated data items on broadcast channels. Retransmission (replication) of data items over a single broadcast channel was investigated in [20]. It was shown that if l is the length of the broadcast, and $q_1 \geq q_2 \geq \ldots \geq q_n$ are the probabilities with which items 1 to n are accessed, respectively, then the number of times, k_i, that each item, p_i, is transmitted should obey the following constraints:

- $\Sigma k_i = l$, where k_i is the number of copies of p_i ,

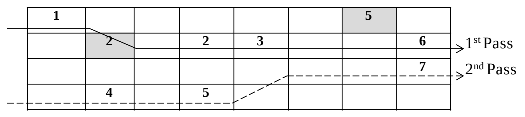
- k_i / k_j should be as close as possible to $\sqrt{q_i} / \sqrt{q_j}$, and
- replicas should be spaced as close as possible to l/k_i from each other on the broadcast channel.

The cyclic retransmission proposed in [20] was extended to multi-channel broadcast in [6]. In this scheme, chunks of the broadcast, denoted as Multiple Broadcast Segments, are replicated. In both aforementioned schemes, replicas are equally spaced to improve response time.

In employing replication to reduce response time and energy consumption, the main challenge faced in generating efficient access schedules lies in selection of the "appropriate" replica. In Fig. 5, a request has been made for retrieval of seven data items. Among the requested data items, it is assumed that data items 2 and 5 have been replicated. If the shaded copies of 2 and 5 are chosen for retrieval, as depicted in Fig. 5(a), then three passes are required to retrieve all requested data items – say data items (1, 3, and 6) in the 1st pass; (2, 5) in the 2nd pass; and (4, 7) in the 3rd pass. However, if the un-shaded copies of 2 and 5 are chosen, as shown in Fig. 5(b), two passes would suffice to retrieve the requested data items – items (1, 2, 3, 6) in the 1st pass and (4, 5, 7) in the 2nd pass. Thus, *the challenge is to identify and retrieve those copies that result in fewer passes and fewer internal switches*.



(a) Shaded copies selected for retrieval – three passes.



(b) Un-shaded copies selected for retrieval – two passes.

Fig. 5: Examples where the choice of replica influences the number of passes.

III. Energy-Efficient Scheduling of Data Retrieval

A. Performance Metrics for Data Retrieval from Parallel Broadcast Channels

The literature has identified three basic performance metrics for the effectiveness of retrieval schemes [8, 12]:

- *Response Time*: the time elapsed from the moment a mobile client generates a request until the requested data items are received.
- *Tune-in Time (active time):* the duration for which a mobile client must remain active to retrieve the requested data items.
- *Channel Switching:* the frequency of channel switches in retrieving the requested data items.

The energy expended for data retrieval can be computed as a function of the aforementioned basic metrics, as follows.

*Energy Consumption = (Response Time − Tune-in Time) * SleepModePower + (Tune-in Time) * ActiveModePower + NumberofSwitches *10% * ActiveModePower*.

According to [14] the power consumption for channel switching is 10% of the power consumed in active mode. We also assumed that the mobile device can operate in either active or sleep mode.

We use response time and energy consumption as performance metrics to evaluate the effectiveness of the proposed data retrieval methods, as these parameters capture both the quality of service and the energy efficiency of data retrieval.

B. Proposed Scheduling Algorithms for Data Retrieval from Indexed Parallel Channels

Recall (from Section II.D) that a matrix $\boldsymbol{C}$, of size $N \times M$ is used to represent a parallel broadcast, where N is the number of channels and M is the number of pages (cells) on a channel. A request may require to access S cells. If the cell $C_{i,j}$ contains a requested data item, then $C_{i,j} \in S$. We represent an *access path*, i.e., the sequence of cells that a mobile device visits during one broadcast cycle, as an array l of length M, where $l\,[j] = i$ signifies that the path accesses cell $C_{i,j} \in S$. Our goal then, given a set S, is to construct a set $L = \{l_1, l_2, \ldots, l_k\}$ with minimum number of access paths, that covers the requested data items.

Before computing the access paths for a user request, one can determine the minimum number of access paths required. This can easily be determined by computing the "cut" of each column: with respect to S, the cut of column j is the number of rows containing requested data items in S that are in either column j or j+1 (note from section II-D, we assumed $OPR=1$). Then the number of access paths required is the maximum of all the columns' cuts. We call this number the *maximum cut*. We will refer to this value as k. We claim that S can be accessed over k broadcast cycles. Our next goal is to generate k access paths with as few internal switches as possible. For this purpose, we introduce the following heuristic based algorithms:

1) Parallel Object Scan (POS) Algorithm

Observe that for each column representing parallel broadcast, an access path visits a cell in some row. We construct L in parallel, starting with column 1 and advancing them based on a few fairly simple and intuitive rules that comprise the *Parallel Object Scan (POS)* algorithm. Assume we have k access paths visiting k rows at column j. Based on this assumption and with respect to S, we advance the k rows to column j+1 according to the following priority rules:

Rule 1: Continue: if a path l is accessing row i in column j (i.e., visiting $C_{i,j}$) and $C_{i,j+1} \in S$ or $C_{i,j+2} \in S$, then this path advance in the same row.

Rule 2: Switch: If $C_{i,j+2} \in S$ for some i, and no path is reading row i in column j, then we must force a path in some other row to switch to row i in column j+1. Among the paths not already accounted for by Rule 1, we choose a path that will not visit a requested data item for the longest time.

Rule 3: Default: Advance remaining paths to column $j+1$.

Repeating this process for all columns (once the initial values for column 1 have been determined) results in a complete solution. What remains to be provided to this algorithm is how the row values for column 1 are determined and how we choose the path that will have to switch in Rule 2.

The initial values of the k access paths (i.e., the row values for column 1) are simply the k rows that contain the first (earliest) requested data items. The choice of rows is not necessarily unique, as there may be more than k rows from which to choose. For example, if $k=3$ and $N=4$ and the earliest requested items appearing on channels are $C_{1,1}$, $C_{2,1}$, $C_{3,3}$, and $C_{4,3}$, then we can choose either rows 1, 2, and 3 or rows 1, 2, and 4. In either case a switch will be required (from row 1 or 2) to one of the other rows. The choice is inconsequential.

A more complex decision arises when applying Rule 2. Assume we have a set of candidate paths from which to choose. As our goal is to reduce the number of internal switches, we should not switch a path that accesses a requested item soon. For example, assume we have two paths reading rows 1 and 2, respectively, in column j, and the next two requested items are $C_{3,j+2}$ and $C_{1,j+3}$. We must switch one of the two paths to row 3 in column $j+1$. If we choose the path currently in row 1, then we will need to switch the other path to row 1 to read the requested item in column $j+3$. Instead, if we choose the path currently in row 2 then we can allow the path currently in row 1 to continue accessing row 1 and no additional internal switch is required. To formalize this choice we need to compute the “distance” along a row until the next requested data item. Our choice for the path to switch is simply the one with the greatest distance from its current column. This concept, and the entire algorithm, is formalized in the Appendix. A running example is also provided to demonstrate the operation of the algorithm.

2) *Serial Empty Scan (SES) Algorithm*

The POS algorithm constructs *k* access paths that cover all the requested items. With respect to the user's request, the *Serial Empty Scan (SES)* algorithm works the other way around; it generates access paths that are needed to access non-requested data items. The basic idea behind the algorithm is as follows: it constructs $N – k$ paths that scan only empty blocks (non-requested data items), iteratively, one at a time. These paths are denoted as *empty paths*. During each iteration, the algorithm also relocates the requested data items into *k* rows, each of which represents a "logical channel." Each of the resulting logical channels represents an access path for the requested data items. The action of relocating data items from one channel to another simulates a switch during a scan.

A logical channel is represented as a row in the broadcast matrix, ***C*** (see Section II.D), and each data item relocated to a logical channel is tagged with its original physical channel (row). We can think of the initial matrix as consisting of *N* logical channels, each corresponding to a physical channel. As the algorithm proceeds, the number of logical channels will be incrementally reduced to *k* by moving blocks (containing requested data items) from one channel to another (containing no requested data items), i.e., an empty block will be incrementally extended until an entire row is empty. Observe that if a channel contains no requested data items, then a trivial empty path is the entire channel. The algorithm finds and eliminates the trivial empty paths immediately. More specifically, the requested data items in cells $C_{i,m}$ through $C_{i,n}$ are moved to cells $C_{j,m}$ through $C_{j,n}$, respectively, as a block containing requested data items is moved from channel *i* to channel *j*. We mark $C_{j,n}$ with a special tag, **SW,** to denote that a physical channel switch occurs at this point in the logical channel. It will always be the case that $C_{i,n}$ does not contain requested data. A row representing a logical channel, then, may contain data that physically occurs on different channels.

The SES algorithm is serial and finds the $N - k$ empty paths, one-by-one, by considering empty blocks. An empty block (of size n) is a sequence of consecutive cells, $C_{i,j}$ to $C_{i,\,j+n-1}$, that contains no requested data items. We label this block as $B_{i,j}$, with $size(B_{i,j}) = n$, and define $end(B_{i,j}) = j+n-1$ (the column of the last cell in the block). Block $B_{u,v}$ *overlaps* block $B_{i,j}$ if $i \neq u$ (blocks occur in different rows) and $v \leq end(B_{i,j}) < end(B_{u,v})$, i.e., block $B_{i,j}$ ends in a column spanned by block $B_{u,v}$. The algorithm constructs an empty path by finding a sequence of *overlapping empty blocks* that includes (at least) one block from each column. For a given column j, a *maximal empty block* of j is a block $B_{i,j}$, such that for any other empty block $B_{k,j}$, $end(B_{i,j}) > end(B_{k,j})$, i.e., the largest empty block that covers column j. Fig. 6 illustrates these notations. A *switching block* is one whose last (rightmost) cell is tagged **SW**. To find a *best block* of column j we find all the switching blocks of that column and then select a maximal one; if there is no switching block, we choose any maximal empty block. As before, we wish to have as few inside switches in our final solution as possible, and this goal corresponds to selecting best blocks as we construct empty paths.

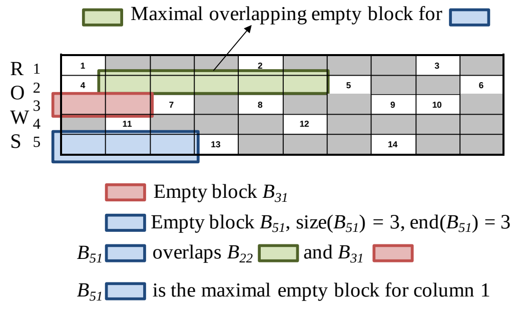

Maximal overlapping empty block for
R O W S
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Empty block B_{31}
Empty block B_{51}, size(B_{51}) = 3, end(B_{51}) = 3
B_{51} overlaps B_{22} and B_{31}
B_{51} is the maximal empty block for column 1

Fig. 6: Notations for SES algorithm.

To begin the construction of an empty path, SES finds a best block for column 1, say $B_{i,1}$. Let $end(B_{i,1}) = n$. It then finds a best (overlapping) block for column n, say $B_{j,n}$. It continues this process until the final empty block ends in column M. What results is an empty path. We can make the following observations about this empty path:

1. Channel *j* must contain at least one requested data item prior to column *n*, else $B_{j,1}$ would have been a longer empty block than $B_{i,1}$.
2. If a block $B_{a,x}$ is followed by block $B_{b,y}$, then cell $C_{a,y}$ contains a requested data item.

These two observations allow us to explain how an empty path that contains at least two empty blocks implicitly determines a non-empty path with an equal number of switches, such that before each switch some requested data item is read. This path begins reading on channel *j*; then whenever the empty path switches *from* some channel, we switch our logical channel *to* that channel. By the above two observations, this logical channel will read a requested data item on each channel that it scans.

To keep track of this logical channel, the algorithm incrementally constructs the logical channel by copying each segment of this channel into the corresponding empty blocks as it is constructing the empty path. So, for example, if the algorithm starts with empty block $B_{i,1}$ followed by $B_{j,n}$, it moves the requested items in cells $C_{j,1}$ through $C_{j,n}$ to channel *i*. Observe that this effectively creates one larger empty block on channel *j*, starting in column 1 and continuing until the end of $B_{j,n}$. The algorithm repeats this relocation each time it finds the next empty block in the path. This process creates a larger and larger empty block, until we have a completely empty channel, which the algorithm then eliminates. As described above, the move operation adds the **SW** tag at the end of each moved block to identify switches and enable the algorithm to correctly choose best blocks during subsequent iterations of the construction of empty paths.

The SES algorithm repeats this procedure $N - k$ times, constructing empty paths, and relocating the remaining requested data items to *k* logical channels. Each of the resulting logical channels represents an access path, containing the channels/data items to be read and indications of where switches occur. It should be noted that moving a block from channel *A* to channel *B* and tagging the last cell **SW** indicates a switch, but subsequently moving the *same* block from channel *B* to channel *C* is not an additional switch ─ it is equivalent to moving data item from channel *A* to channel *C*. Thus, the strategy of choosing blocks tagged **SW** minimizes the number of inside switches. Choosing the maximal empty block at each step also contributes to minimizing the number of inside switches during a scan. A running example of the operation of the SES algorithm is provided in the Appendix.

C. Enhancements to the POS and SES Algorithms

The POS and SES algorithms focus on reducing the number of inside switches and minimizing the number of passes required. We now consider additional issues, such as the number of outside switches. These issues typically have less significant effects on the number of passes since it happens between two successive broadcast cycles. Nevertheless, some intuitive observations and modifications could improve our targeted performance metrics.

1) Minimizing Outside Switches

Observe that while both the POS and SES algorithms produce access paths, neither specifies the order in which the access paths should be applied. Note further that the algorithms offer some degree of freedom, to allow arbitrary choice between the set of available rows. This non-determinism leads to multiple solutions with the same number of inside switches, but a possibly different number of outside switches. Given a set of access paths, we wish to order them, in an attempt to minimize the number of outside switches and hence the overall energy consumption.

To minimize the number of outside switches, we need to order the set of access paths such that whenever possible, the ending channel (row) of one access path coincides with the starting channel (row) of the next access path. To do this, we construct a graph and extract a minimum spanning tree (MST) to yield an ordered set that minimizes the number of outside switches.

Given a set of access paths, we construct a weighted, directed, complete graph $G \equiv (V, E)$, in which nodes represent the access paths, i.e., $V_l \in V$ corresponds to access path l. The directed edge between every pair of nodes (l_i, l_j) is weighted 0 if l_i ends in the same channel from which l_j starts, otherwise the weight is 1. This process generates a complete weighted graph, with every node having at most one outgoing edge weighted 0. From this graph, we construct an MST in which every node (except one) has exactly one child. This gives the order in which the access paths should be processed. Since the tree will have minimum weight, we will have the minimum number of outside switches.

2) Reducing Response Time at the Expense of Increasing Energy Consumption

The proposed POS and SES algorithms find the minimum number of passes required for retrieval of all the requested data items, which reduces energy consumption, but neither algorithm guarantees the shortest possible response time.

For a given solution (sequence of access paths), the response time could be shortened if the last access path reads the last data item as early as possible. This result, which may be achieved at the expense of increasing the number of inside switches in some of the access paths, is the objective of the *response-time POS (rPOS) scheduling algorithm*, a modified version of the POS algorithm. Observe that the earliest point (smallest column number) at which the last access path can read its last requested data item is at the rightmost column that has $cut = k$.

Since the minimum number of passes is k, then deciding between energy consumption or response time necessitates that we determine the energy consumption corresponding to k, $k+1$, $k+2$, …, N passes.

The *energy-consumption-oriented SES (eSES)* algorithm, an extension of the SES algorithm, is designed to allow the user to trade off these two performance metrics against each other. Recall that the SES algorithm starts with N access paths and reduces the number of paths by one after each pass through the matrix $\boldsymbol{C}$, which represents the parallel broadcast channels (see Section II.D). The eSES algorithm records the number of inside switches and the energy consumption for each iteration. The user can then select a desired schedule, trading off energy consumption for response time (viz. number of access paths).

IV. Replication

As noted in Section II.E, *replication* could be used to potentially reduce the number of passes and energy consumption. Specifically, data items commonly requested can be reproduced and broadcast on more than one channel to increase their availability. The expense incurred is an overall increase in the length of the broadcast, which will increase the response time and energy consumption associated with the retrieval of non-replicated items. This highlights the importance of prudent selection of the data items to be replicated.

In cases when data replication is allowed, a data item o_i can be uniquely represented as a triplet $<id_i,c_i,p_i>$, where id_i is the ID of o_i, c_i is the channel (row) ($1 \le c_i \le N$) on which the data item is broadcast, and p_i is the page (column) ($1 \le p_i \le M$) on which it appears. Assume OID and $O = \{<id_i, c_i, p_i>\}$ represent the set of IDs and the set of data items on the broadcast channels, respectively. Let $S_{id} = \{id_i \mid id_i \in OID\}$ and $S = \{<id_i, c_i, p_i> \mid id_i \in \mathrm{S_{id}}\}$ represent the IDs and the set of requested data items, respectively. Finally, let $S_R = \{<id_i, c_i, p_i> \mid id_i \in S_{id}\}$ ($S_R \subseteq S$) and $S_{NR} = S - S_R$ be the set of replicated and non-replicated requested data items, respectively.

The problem at hand is to select a set, S'_R, such that:

- $S'_R = \{<id_i, c_i, p_i> \mid <id_i, c_i, p_i> \in S_R$ AND for any $i \neq j$, $id_i \neq id_j\}$, i.e., S'_R contains exactly one copy of every data item 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.

We proposed two heuristic-based algorithms. These algorithms retain the copy of requested replica that results in a reduced number of passes and channel switches, respectively. These algorithms logically "remove" replicas that violate the aforementioned criterion from the broadcast, and then call on the scheduling algorithm, e.g., POS or SES, to generate the access paths.

A. *The Greedy MAX_cut Algorithm for Reducing the Number of Passes*

The Greedy MAX_cut Algorithm (GMA) is intended to reduce the number of passes (see Appendix). The running example in Figs. 7(a)-(e) is used to illustrate the operation of the GMA. From Fig. 7(a), the initial number of required passes is three, equal to the maximum cut. The GMA iteratively reduces the initial number of passes, one pass at a time, according to the following steps:

Step 1: Mark removable data items in the broadcast: replicated data items in maximum-cut column pairs are identified and marked as *'removable'* (Fig. 7(b)). Note that in two cases the replicated data items are not marked as removable:

a. Removal of the replica has no effect on the MAX_cut, i.e., a replicated item in a maximum-cut column has a non-replicated item as a neighbor and hence its removal makes no difference to the MAX_cut. For instance, removing item <9, 2, 3>, shaded in Fig. 7(a), makes no difference to the cut at the critical column 2, and hence to the MAX_cut.

b. A replicated data item has just one more copy, both copies are adjacent to each other, and one of the copies is in a maximum-cut column. In this case removal of the data items *as a pair* can potentially reduce the cut. However, the data items cannot be marked as removable, since at least one copy must remain in the broadcast matrix.

Note that if there is even one maximum-cut column that does not have removable data items, then MAX_cut cannot be reduced. Let *R* be the set of removable data items.

Step 2: Remove pairs of adjacent data items from *R* i.e., data items 2 and 8 in the column pair {8, 9} (shaded in Fig. 7(b)). In this step, the algorithm first removes those replicas that are the *only* removable items (Fig. 7(d)).

Note, it is possible that a data item that was previously marked as removable now becomes the only copy; such a data item is to be unmarked; refer to Fig. 7(b); removing <2, 3, 8> forces the unmarking of <2, 3, 6> (shaded in Fig. 7(c)), as it becomes the only remaining copy of data item 2. Fig. 7(c) depicts our running example after this step.

Step 3: Least-Cut rule: *In case of multiple choices for removal, the algorithm chooses a data item that has a replica that occurs in a pair of columns with the least cut in the matrix* (Fig. 7(d)).

Step 4: Identify the new maximum-cut column pairs and repeat steps 1-3: Each time the MAX_cut is reduced, a new set of maximum-cut column pairs is generated.

B. The Greedy Block Algorithm for Reducing the Number of Internal Switches

The Greedy Block Algorithm (GBA) aims at reducing the number of internal switches, and hence reducing the energy consumption. After the primary optimization criterion (number of passes, Section IV.A) the GBA is run to satisfy the secondary optimization criterion (number of internal switches). Upon application of the GMA, we are left with a set of replicas, S_{GMA} , such that $S_{GMA} \subseteq S_R$. Application of the GBA results in the set S_R', such that $S_R' = \{<id_i, c_i, p_i> | <id_i, c_i, p_i> \in S_R$ AND for any $i \neq j$, $id_i \neq id_j\}$, i.e., S_R' contains exactly one copy of every data item in S_R. The heuristic used by the GBA for selecting a copy among many copies of a replicated data item follows the largest empty block rule:

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* (Fig. 7(e)). The intuitive idea behind the largest empty block rule is that by removing data items that result in the formation of large empty blocks, we attempt to select those replicas that are closer to other data items on a channel. This increases the chance that these copies are read in the same broadcast cycle without the need for additional switches. In short, if a replicated data item is flanked by large blocks, removing it could reduce the number of switches.

In every step, the GBA computes all "potential blocks" and removes the largest among them. If two potential blocks are of the same size, then the rightmost block rule is applied:

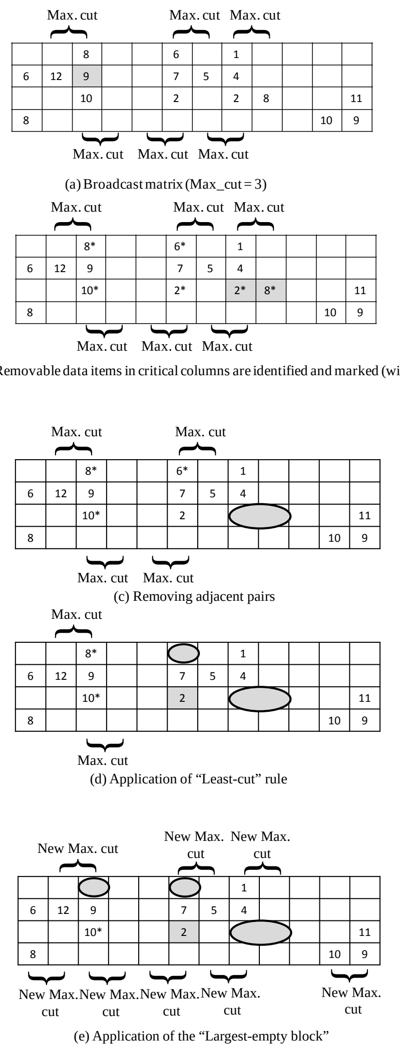
Rightmost block rule*: If two (or more) potential blocks have the same size, then the "rightmost" block is chosen for removal.*

By retaining replicas that occur earlier, the rightmost block rule tries to mimic the behavior of the *rPOS* algorithm (see Section III.C.2), i.e., to end the retrieval process earlier.

V. Simulation Results

A. Simulation Design

A simulator was developed to verify the behavior of the proposed algorithms and measure their effectiveness against the heuristic approach proposed in [10]. For the sake of simplicity, without loss of generality, the simulator simulates the indexing layout depicted in Fig. 2(a). Based on our experimental results, to compensate for the execution time of each algorithm, a delay gap of one page is inserted between the index and data items to avoid missing any requested data items in the first broadcast pass. The simulator views the parallel wireless channels as a two-dimensional $N \times M$ array, where N and M represent the number of parallel wireless channels and the number of data items on a broadcast, respectively. For each simulation run, user requests are generated requesting K random data items on the broadcast.



(a) Broadcast matrix (Max_cut = 3)

(b) Removable data items in critical columns are identified and marked (with *)

(c) Removing adjacent pairs

(d) Application of “Least-cut” rule

(e) Application of the “Largest-empty block”

Fig. 7: Running example for the Greedy MAX_cut algorithm.

The NASDAQ database with 4290 securities is used as the source data for the data items on the broadcast. The simulator assumes a number of 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 section.

B. Simulation Results

Conceptually, the POS and SES algorithms 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.

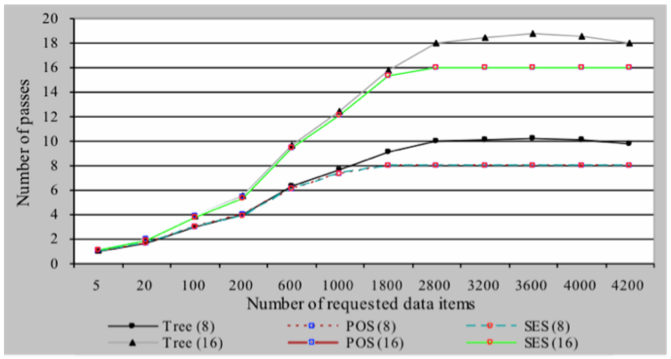
TABLE 1
INPUT PARAMETERS FOR THE SIMULATOR

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

1) Number of Passes

Fig. 8 shows the number of passes for the POS, SES, and the heuristic proposed in [10] (identified as “Tree” in Fig. 8). As expected, regardless of the number of data items requested and the physical configuration of the simulated environment, the POS and SES algorithms generate the same number of passes.

When the number of requested data items, *K*, is relatively small, the “Tree” algorithm is as efficient as the POS and SES algorithms. However, as *K* grows larger, the “Tree” algorithm becomes relatively less efficient. Interestingly, relative to the number of wireless channels (*N*), after a threshold value, “Tree” requires more passes than the *Row Scan* algorithm. Increasing *K* and/or *N* increases the number of passes required to pull requested data items from the parallel wireless 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 wireless channels. As *K* approaches ((*M**(*N*–1)) + 1), *MAX_cut* approaches *N*. At this threshold point, the POS and SES algorithms behave similar to the *Row Scan* algorithm, as it would be the most cost-efficient algorithm to use.



Number of channels = 8 or 16

Fig. 8: Number of broadcast passes.

2) Number of Channel Switches

Fig. 9 shows the number of inside channel switches for different scheduling algorithms. As expected, the POS and SES algorithms require almost the same number of switches, regardless of the number of channels and/or the number of data items requested. Compared to these two algorithms, “Tree” requires a considerably greater number of channel switches, especially as *K* grows larger. Finally, relative to the POS and SES algorithms, rPOS requires a few more switches, confirming the tradeoff anticipated between energy consumption and response time.

As *N* and/or *K* increases, more inside switches are needed, as typically a greater number of requested data items are distributed over different channels. When *K* increases beyond a threshold, the number of channel switches begins to decrease. This is because the number of passes approaches *N*. As *K* approaches ((*M**(*N*–1)) + 1), the number of inside switches goes to zero. As noted earlier, at this threshold, the *Row Scan* algorithm, which requires no inside channel switches, would be applied.

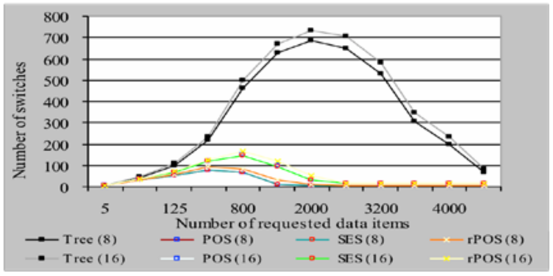


Fig. 9: Number of inside switches.

Fig. 10 shows the number of outside switches required for the different scheduling algorithms. As expected, the POS, SES, and rPOS algorithms require fewer outside switches than the "Tree" algorithm, especially when 20-30% of the data items is requested. When the number of requested data items approaches the threshold, the number of outside switches increases drastically and approaches (*N*–1). Fig. 10 also shows that relative to the POS and SES algorithm, as the number of requested data items increases, the number of outside switches for rPOS increases more smoothly.

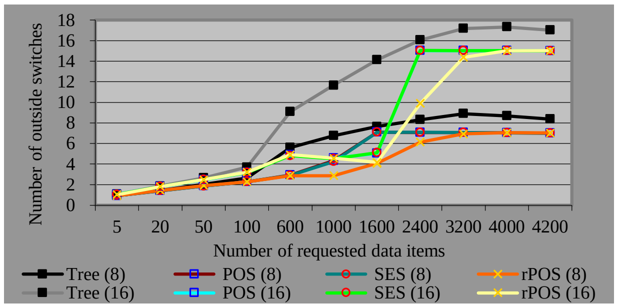
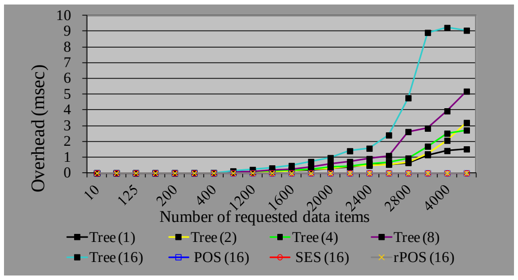


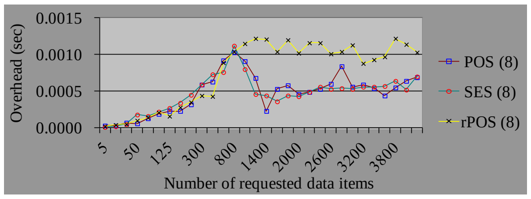
Fig. 10: Number of outside switches.

3) Cost of Algorithms

It was one of our objectives to develop a scheduling algorithm that generates access paths with reasonable overhead cost (complexity). Fig. 11(a) shows the overhead cost of different scheduling algorithms. As noted in [10], the overhead cost of the "Tree" algorithm is O(K^2), much more than the cost of POS or SES algorithms, especially when a large number of data items is requested. The overhead cost of the POS and the SES algorithms is O(N^2M) in the worst case, as shown in Fig. 11(b). From this figure, one can conclude that the overhead cost of the POS and SES algorithms linearly increases when a relatively small number of data items is requested. However, as the number of requested data items increases, the overhead cost decreases, since the aforementioned algorithms behave more and more as the *Row Scan* algorithm. After the threshold point, the cost remains constant, since the *Row Scan* algorithm is always applied. Finally, the rPOS algorithm, as expected, is relatively more expensive, due to the additional computational overhead.



(a) Overhead cost of Tree-based, POS, SES, and rPOS algorithms.



(b) POS, SES, and rPOS algorithms (Number of channels = 8).

Fig. 11: Overhead cost of different Scheduling algorithms.

4) *Response Time*

Fig. 12 shows the response time of the 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” algorithm usually requires a longer response time, especially when *K* is large. Interestingly, simulation results showed that for small *K*, the POS and SES algorithms have slightly longer response times than the “Tree” algorithm. This is due to the heuristics employed by “Tree”, since it retrieves as many data items as possible in earlier passes.

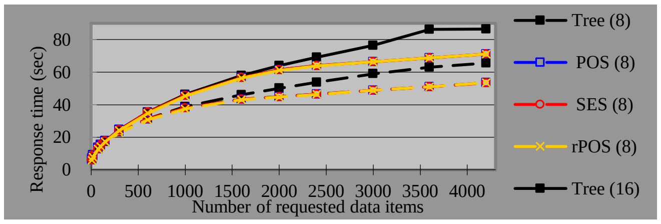


Fig. 12: Response time (number of channels = 8 and 16).

5) Energy Consumption of Data Retrieval

Fig. 13 shows the energy consumption of the different scheduling algorithms. The total energy consumption is determined based on the consumed energy in active and doze modes, and in the channel switching phase. Obviously, for different scheduling algorithms, the consumed energy in active mode is almost the same. Similarly, the energy consumed in doze mode is not very different for different scheduling algorithms. As a result, one can conclude that the energy consumed in the switching phase is the dominating factor that distinguishes different scheduling algorithms. As a result, the shapes of the curves in Fig. 13 are similar to those of Fig. 9, which depicts the number of inside switches.

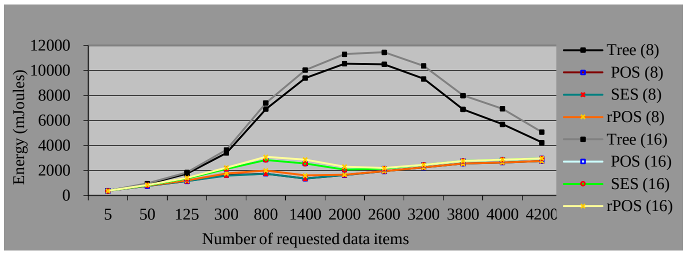
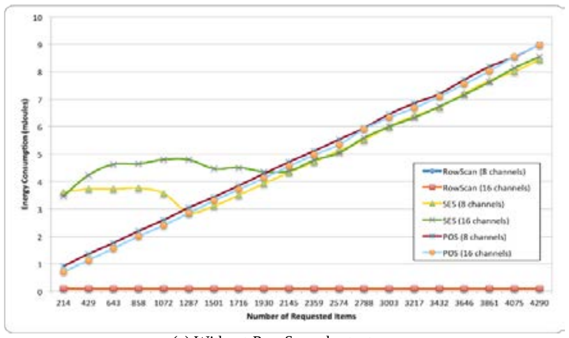


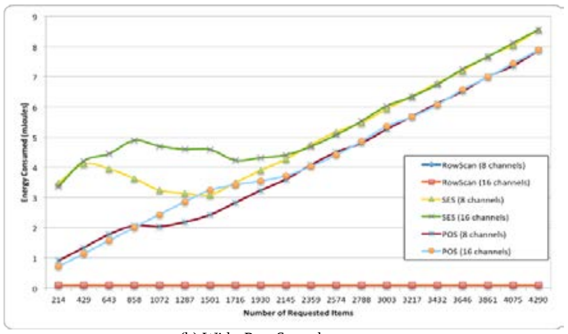
Fig. 13: Energy consumption of data retrieval algorithms.

6) *Energy Consumption of Scheduling Algorithms*

We ran the simulator on an emulated, ARM-based mobile device, while a profiling tool recorded each instruction executed by the emulated processor. Histograms of executed operations were generated for simulated scheduling algorithms. Based on the average energy consumption of different ARM operations [4], the histograms were used to estimate the energy consumption of each algorithm (Fig. 14(a)). As expected, *Row Scan* algorithm consumes the least energy, as it is essentially a nested for-loop. However, *POS* and *SES*, incur higher energy consumption as the number of requested items increases. The initial high energy consumption of *SES* relative to *POS* is due to the overhead of manipulating logical broadcasts. Additionally, the similar growth of *SES* and *POS* is due to their shared use of *Max_cut*.



(a) Without *Row Scan* shortcut



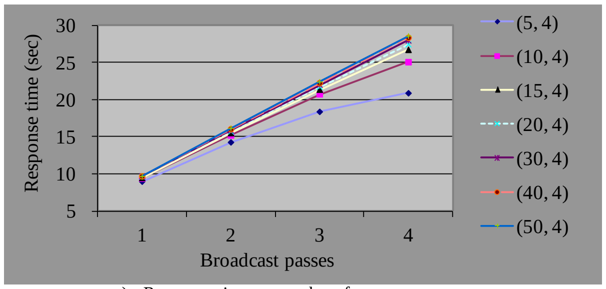
(b) With *Row Scan* shortcut

Fig. 14: Energy consumption of scheduling algorithms.

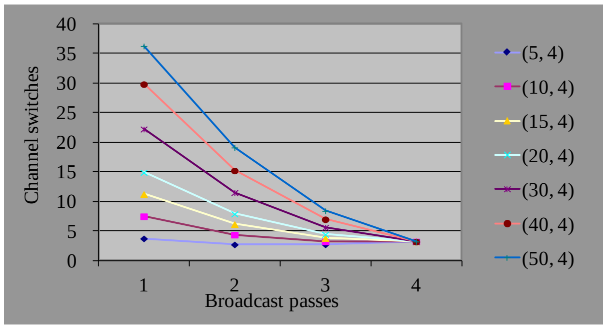
Fig. 14(b) shows how energy can be saved by short-cutting *POS* and *SES* when the number of channels is equal to the *Max_cut* of a broadcast. When each channel requires its own access path, it makes sense to use *Row Scan* instead of a more complicated algorithm like *POS* or *SES*. Though this shortcut saves some energy, computing the maximum cut still incurs energy costs. Despite this cost, the energy saved by using smarter scheduling algorithm outweighs the energy saved by using simpler algorithms.

7) *Trade-off between Broadcast Passes and Channel Switches*

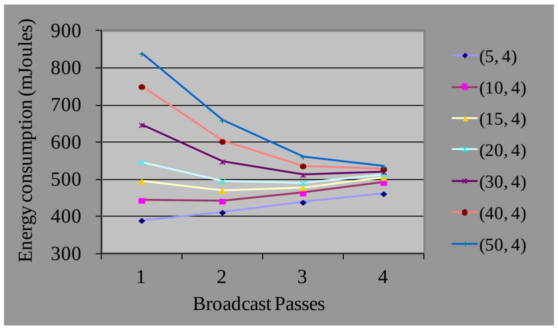
There is a trade-off between the response time and the number of channel switches (energy consumption). To confirm this, we ran the simulator for a fixed configuration, i.e., the same number of requested data items and the same number of channels, and observed the number of channel switches and the corresponding response time. Results are depicted in Fig. 15. Each subfigure plots the performance for different numbers of requested items and number of broadcast channels. As can be seen, a greater number of passes results in longer response times and fewer channel switches. On the other hand, the energy consumption depends on the number of passes and the number of channel switches. Fig. 15(c) illustrates that the number of data items 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 data items, energy consumption can be reduced by taking a smaller number of passes, whereas for a relatively large number of requested data items, taking a greater number of passes over the parallel channels results in lower energy consumption. Between these two boundaries, a medial number of passes results in the lowest energy consumption.



a) Response time vs. number of passes.



b) Channel switching vs. number of passes



c) Energy consumption vs. number of passes.

Fig. 15: Response time, number of switches, and energy consumption vs. number of passes.

C. Effect of Replication

To show the effectiveness of data replication, some data items were randomly chosen as hot data, and the simulator for the POS algorithm was extended to mimic the behavior of the GMA and GBA algorithms. As in our earlier simulation runs, user requests were generated, the GMA and GBA algorithms were applied, and finally the POS algorithm was called to generate the access paths. Fig. 16 shows the effect of data replication on the number of passes when replicas are randomly distributed among the channels. As anticipated, in general, data replication reduces the number of passes, regardless of the number of the data items requested.

In another simulation run, the simulator measured the effect of data replication on the response time, where replicas were distributed based on the protocol proposed by Wong in [20]. Fig. 17 shows the results. Again, as expected, data replication has a direct effect on the response time, regardless of the number of items requested.

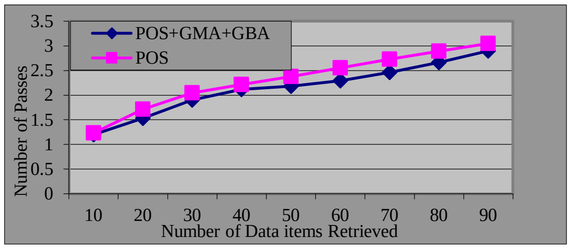


Fig. 16: Effect of data replication on the number of passes (number of channels = 10).

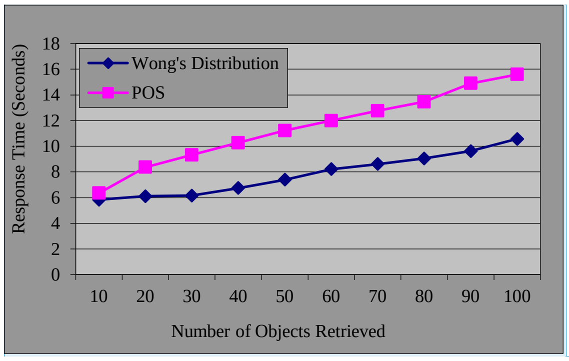


Fig. 17: Effect of data replication on response time (number of channels = 10).

Commented [A1]: Resized the legend, so that “POS” shows up.

Data replication also affects the number of inside switches, and hence the energy consumption. In a separate simulation run, the simulator measured the number of inside switches for the POS algorithm, with and without the application of the GMA and GBA algorithms. Fig. 18 depicts the results. As expected, replication has significant impact on the number of inside switches.

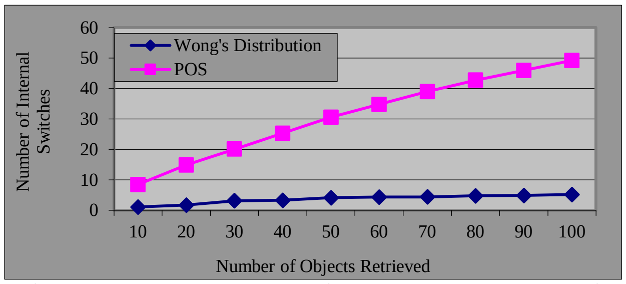


Fig. 18: Effect of data replication on number of inside switches (number of channels = 10).

VI. Conclusion and Future Directions

This paper discussed data retrieval in an indexed, parallel-broadcast wireless channel environment. Several scheduling algorithms, namely Parallel Object Scan (POS) and Serial Empty Scan (SES), were presented. The proposed algorithms were simulated and compared to the heuristic model proposed in [10], using performance metrics such as energy consumption, response time, and number of channel switches. Simulation results showed that the proposed scheduling algorithms generate access paths that have the minimum number of passes and fewer inside switches, at a reasonable overhead cost.

In addition, to reduce the response time and the number of outside switches, a scheduling algorithm was also proposed for optimizing the access paths generated by the POS and SES algorithms. Moreover, two algorithms, namely, rPOS and eSES, were introduced to trade off response time and energy consumption. The rPOS algorithm can always generate access paths with the shortest response time, at the expense of higher energy consumption than the POS and SES algorithms. On the other hand, the eSES algorithm can yield a reduced energy consumption for a given response time. Moreover, the POS and SES algorithms were emulated to measure their energy consumption.

Finally, we investigated the effect of replication and developed two algorithms that attempt to choose the replicas that reduce both the number of passes and the number of internal switches.

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

- Scope of the work can be extended for channels with different bandwidths.
- Scope of the proposed algorithms can be extended to allow mobile devices to more intelligently operate in various operational modes.
- To further reduce response time and energy consumption, one can develop a scheme that locally caches the index in the mobile devices.
- The simulator can be extended to find the channel layout (relative positions of index and data items) that offers the best performance (refer to Fig. 2).
- User profiles can be used to bundle multiple potential queries in order to reduce the overall energy consumption.
- The tree-based indexing scheme has some weaknesses in retrieving data items based on multiple attributes. It is possible to combine the strengths of signature-based indexing with the strengths of tree-based indexing, toward the goal of reducing energy consumption and response time in retrieving data items based on multiple attributes.
- Currently, we are studying the application of broadcasting as a means to facilitate traffic prediction [13].

REFERENCES

[1] S. Acharya, and M. J. Franklin, and S.B. Zdonik, “Disseminating Updates on Broadcast Disks,” *Proceedings of the 22nd International Conference on Very Large Data Bases*, 1996.

[2] B. Aktus and A. Bria, “Evaluation of user perceived performance in sparse infrastructure wireless systems,” *Proceedings of the 2nd Annual Workshop on Affordable Wireless Services and Infrastructure*, 2004.

[3] E. Ardizzoni, A. A. Bertossi, M. C. Pinotti, S. Ramaprasad, R. Rizzi, and M. V. S. Shashanka, “Optimal skewed data allocation on multiple channels with flat broadcast per channel,” *IEEE Transactions on Computers*, 2005, 54(5):558-572.

[4] M. Bazzaz, M. Salehi, and A. Ejlali, "An Accurate Instruction-Level Energy Estimation Model and Tool for Embedded Systems," *IEEE Transactions on Instrumentation and Measurement*, 2013, 62(7):1927-1934.

[5] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th Edition, Pearson, 2016.

[6] C. Hsu, G. Lee, and A.L. Chen, “Index and Data Allocation on Multiple Broadcast Channels Considering Data Access Frequencies,” *Proceedings of the Third International Conference on Mobile Data Management*, 2002.

[7] 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*, 2004, 16(9):1143-1156.

[8] Q.L. Hu, and D.L. Lee, "A Hybrid Index Technique for Power Efficient Data Broadcast," *Distributed and Parallel Databases Journal*, 2001, 9(2):151-177.

[9] J.Hu, R German, A. Heindl, R. Kates, and M. Unbehaun, "Traffic modelling and cost optimization for transmitting traffic messages over a hybrid broadcast and cellular network," *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems, 2005.*

[10] A.R. Hurson, A.M. Muñoz-Avila, N. Orchowski, B. Shirazi, Y. and Jiao, "Power-Aware Data Retrieval Protocols for Indexed Broadcast Parallel Channels," *Journal of Pervasive and Mobile Computing*, 2006, 2(1):85-107.

[11] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Data on Air: Organization and Access", *IEEE Transactions on Computers*, 1997, 9(3):353-372.

[12] J. Juran, A.R. Hurson, and N. Vijaykrishnan, "Data Organization and Retrieval on Parallel Air Channels: Performance and Energy Issues," *ACM Journal of WINET*, 2004, 10(2): 183-195.

[13] L. Heendaliya, M. Wisely, D. Lin, S. Sedigh Sarvestani, and A.R. Hurson, "Influence-Aware Predictive Density Queries Under Road Network Constraints," 14th International Symposium on Spatial and Temporal Databases, August 2015.

[14] G.L. Lee and S.C. Lo, "Broadcast data allocation for efficient access of multiple data items in mobile environments," *ACM Mobile Networks and Applications*, 2003, 8(4):365-375.

[15] G. Li, Q. Zhou, and J. Li, "A Novel Scheduling Algorithm for Supporting Periodic Queries in Broadcast Environments", *IEEE Transactions on Mobile Computing*, 2015.

[16] W. Marcel, K. Lange, M. Baumann, and M. Weber, "Autonomous Driving: Investigating the Feasibility of Car-Driver Handover Assistance," *7th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2015:11–18.

[17] MSN Direct, *Available* at http://www.msndirect.com/about-msn-direct.aspx, retrieved June 2009.

[18] W. Sun, C. Chen, B. Zheng, C. Chen, and P. Liu, "An Air Index for Spatial Query Processing in Road Networks", *IEEE Transactions on Knowledge and Data Engineering*, 2015:382–95.

[19] L. Wischhof, A. Ebner, and H. Rohling, "Self-Organizing Traffic Information System based on Car-to-Car Communication: Prototype Implementation," *Proceedings of the 1st International Workshop on Intelligent Transportation (WIT 2004)*, 2004.

[20] J. Wong, "Broadcast Deliver," *Proceedings of the IEEE*, 1998, 76(12):1566-1577.

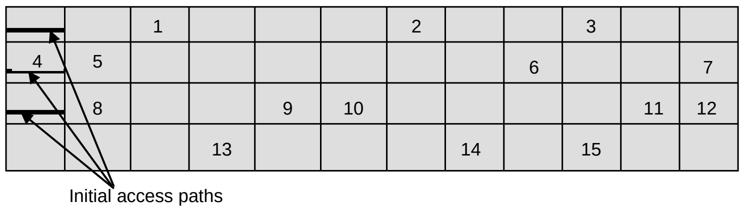
[21] W. G. Yee, S. B. Navathe, E. Omiecinski, and C. Jermaine, "Efficient data allocation over multiple channels at broadcast servers," *IEEE Transactions on Computers*, 2002, 51(10):1231-1236.

[22] J. Zhang and L. Gruenwald, "Efficient Placement of Geographical data over broadcast channel for spatial range query under quadratic cost model," *MobiDE*, 2003: 30-37.

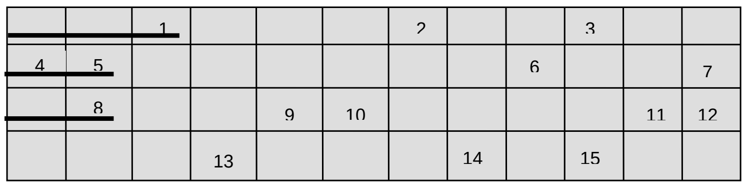
[23] B. Zheng, X. Wu, X. Jin and D.L. Lee, "Broadcast scheduling: TOSA: a near-optimal scheduling algorithm for multi-channel data broadcast," *Proceedings of the 6th International Conference on Mobile Data Management*, 2005: 29-37.

Appendix

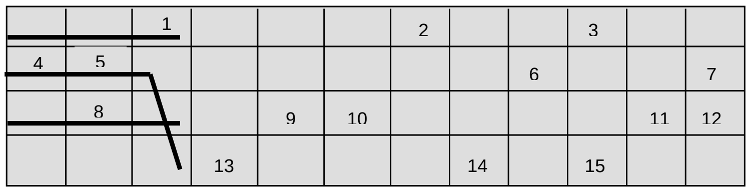
Fig. A1 is a running example that shows the operation of the POS algorithm in finding the optimal number of access paths for the requested data items. The example shows 15 data items requested from a four-channel parallel broadcast configuration. Computing the cut for each column, we determine that the maximum cut is three, and hence, three access paths are sufficient to retrieve the requested data. To initialize the paths in column 1, we see that rows 1, 2, and 3 are the three paths that contain the earliest requested items (Fig. A1(a)). Computing the rows to access in column 2, we apply Rule 1 to all three paths (Fig. A1(b)). Computing the rows to access in column 3, we apply Rule 1 to the path in row 1 (since there is a requested item $C_{1,3}$). We then apply Rule 2, since there is a requested data item $C_{4,4}$. Considering the two remaining paths (not yet assigned a row for column 3), the path reading row 2 has the greater distance (next requested item in row 2 is in column 9) so it is the path switched to row 4. We apply Rule 3 to path 3 since there are no other requested data items in columns 3 or 4 (Fig. A1(d)). The remaining columns are filled similarly (Fig. A1(e)).



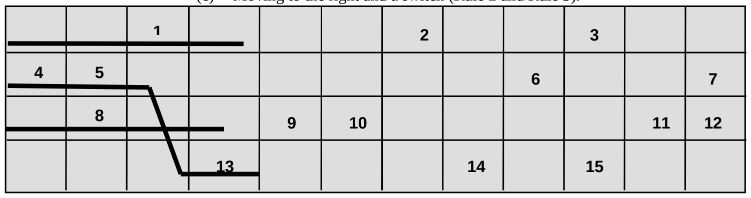
(a) Access paths are initialized.



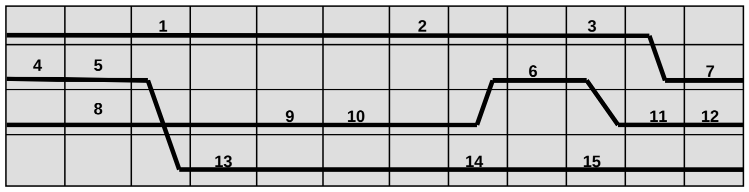
(b) Extending the access paths without channel switching (Rule 1).



(c) Moving to the right and a switch (Rule 2 and Rule 3).



(d) Advancing the access paths without switching (Rule 3).

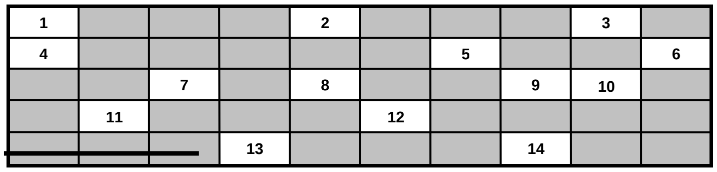


(e) Generation of the final access paths.

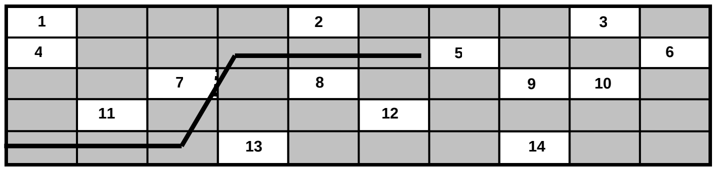
Fig. A1: A running example of the POS algorithm.

A. Serial Empty Scan (SES) Algorithm

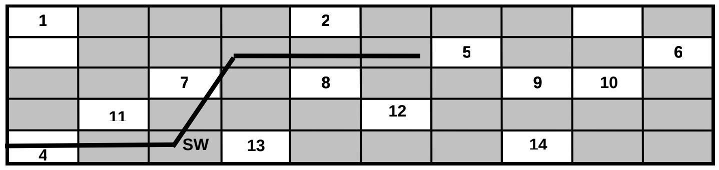
Fig. A2 depicts snapshots of a running example of the SES algorithm, where the goal is to retrieve 14 data items from five parallel channels. For this request, *k* is three, *N* is five, and at least three passes, with at least seven inside switches, are required to retrieve the requested data items.



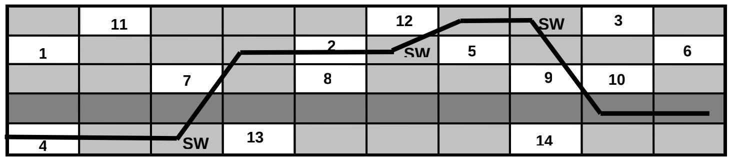
(a) Starting with the largest leftmost empty block.



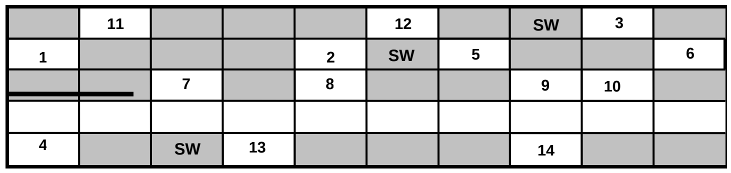
(b) Find an overlapping empty block and switch.



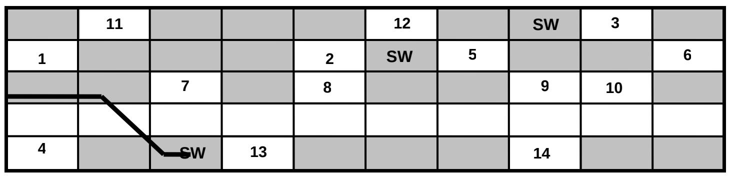
(c) Logical rearrangement of object 4 and advancing the empty block.



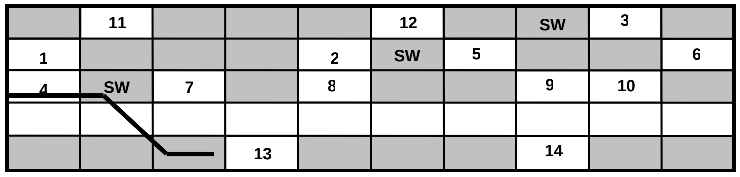
(d) End of first iteration, and generation of an empty channel, in this case, channel 4.



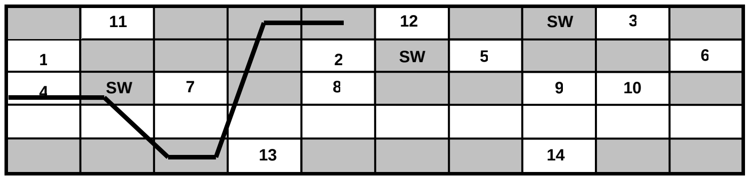
(e) Second iteration, starting with the largest leftmost empty block.



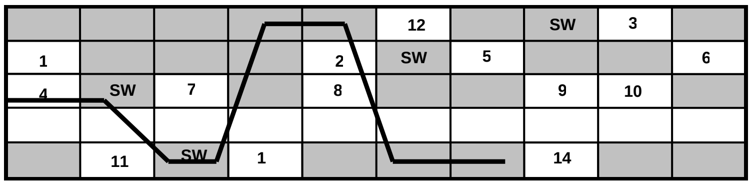
(f) Find an overlapping empty block and switch.



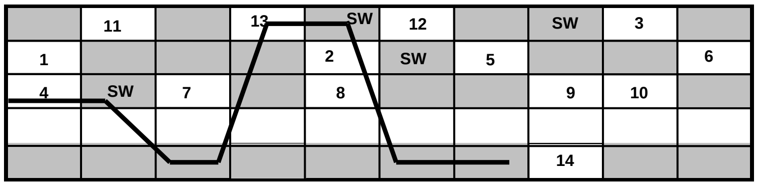
(g) Logical rearrangement of object 4 and advancing the empty block.



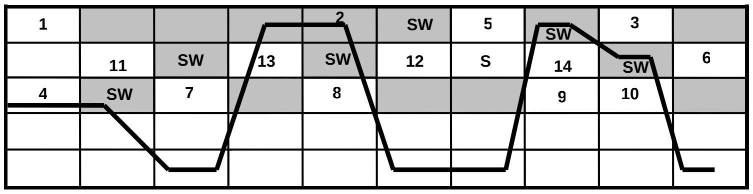
(h) Find an overlapping empty block and switch.



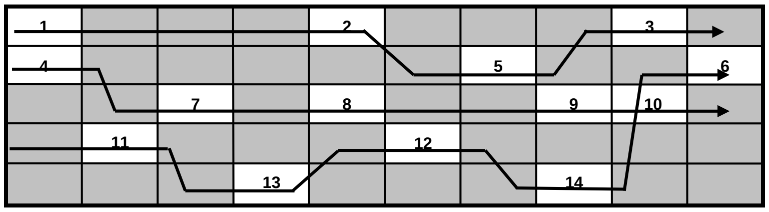
(i) Logical rearrangement of object 11, advancing the empty block, and finding an overlapping empty block and switch.



(j) Logical rearrangement of object 13.



(k) Completion of the second iteration and generation of second empty channel.



(l) The final access paths.

Fig. A2: A running example of the SES algorithm.

As illustrated in Fig. A2, in the first pass, data items 1, 2, 5, and 3 are retrieved. During the second pass, data items 4, 7, 8, 9, and 10 are fetched. Finally, during the third pass, data items 11, 13, 12, 14, and 6 are retrieved, completing the set of requested data items.