
Multimedia Tools and Applications 9, 69–94 (1999)
© 1999 Kluwer Academic Publishers. Manufactured in The Netherlands.

Object Organization on a Single Broadcast Channel in the Mobile Computing Environment

Y.C. CHEHADEH chehadeh@cse.psu.edu
A.R. HURSON hurson@cse.psu.edu
Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802

M. KAVEHRAD kavehrad@engr.psu.edu
Department of Electrical Engineering, The Pennsylvania State University, University Park, PA 16802

Abstract. Advances in computation and communication technologies allow users to access computer networks, using portable computing devices via a wireless connection, while mobile. Furthermore, multidatabases offer practical means of managing information sharing from multiple preexisting heterogeneous databases. By superimposing the mobile computing environment onto the multidatabase system, a new computing environment is attained.

In this work, we concentrate on the effects of the mobile-computing environment on query processing in multidatabases. We show how broadcasting, as a possible solution, would respond to current challenges such as bandwidth and storage limitations. Organizing data objects along a single-dimension broadcast channel should follow the semantic links assumed within the multiple-dimension objects' structure. Learning from our past experiences in objects organization on conventional storage mediums (disks), we propose schemes for organizing objects along a single broadcast air channel. The proposed schemes are simulated and analyzed.

Keywords: mobile computing, wireless communication, multidatabases broadcasting, global information sharing

1. Introduction

Traditionally, most computing environments assume a networked architecture and applications based on client-server or peer-to-peer paradigms. The widespread availability of computer networks and the popularity of portable computing devices have made it possible to access information resources regardless of the location. However, the user is limited in accessing information resources while he/she is physically connected to a network—"sometime/somewhere" environment. Recently, we have witnessed a new breed of application in which, regardless of the location, the user is required to retrieve, update, and process information while in motion and without an explicit physical connection to the network—"anytime/anywhere" or location-independent environment [8]. The mobile computing environment (ubiquitous or nomadic) was developed to meet these new requirements [11, 12, 19]. Three main characteristics, namely, wireless communication, mobility, and portability, distinguish this environment [12]. As reported in the literature, frequent disconnection, lower communication bandwidth, limited energy sources and limited resources at the mobile host unit are current challenges facing the mobile computing environment.

In parallel to the recent technological advances, information technology has also experienced a great deal of progress and maturity during the past three decades. The so called

multidatabases and federated databases were intended to facilitate the global information sharing process among a collection of autonomous and heterogeneous databases in spite of their syntactical and semantic differences. With the networked environment as the underlying infrastructure, multidatabase systems (MDBSs) provide a means for integrating the data from preexisting, distributed, heterogeneous databases and presenting global users with transparent methods to access the global information with a single query [5, 18]. A key feature is that the individual databases retain their autonomy, in order to serve their existing customer set. Such preservation of local autonomy protects an organization's existing investment and users' training.

A new computational environment is obtained by superimposing the mobile computing paradigm over the traditional multidatabase environment. This new environment differs from the traditional multidatabase environment by the fact that it supports a new breed of client. These clients are mobile, communicate in a wireless format, and contain limited resources and energy sources [2, 8]. The challenges introduced by this computing environment have their root in the characteristics of the mobile devices and thus influence different aspects of multidatabases, e.g., query processing. Therefore, it is necessary to find alternative solutions to counteract such impact. The literature has suggested broadcasting (one-way communication pattern) as a possible solution to the traditional two-way communication scheme. In broadcasting, the data (public data, e.g., stock prices) is broadcast over the air channels. A mobile host unit in search of certain data could access the appropriate air channel upon which the data is being broadcast, and retrieve the required data items.

The format and structure of the data on the air channel dictate the method by which the search and retrieval operations are performed. For numerous applications, the data to be broadcast is of a multimedia or hypertext format (text, graphics, video, and voice). Such data, and in most cases, would originate from an information resource available within the federation of information that is normally governed by a multidatabase system. Object-oriented modeling has proven to be effective in modeling multimedia data and integrating data from multiple resources [3, 13]. Therefore, in this work, we assume the data on the air channel to appear as objects.

Object clustering as a means of improving performance has been studied extensively within the scope of conventional object-oriented databases [4, 7, 9, 15, 16]. Due to the natural differences between the serial air channel and the random-access disk, one has to look at different and efficient methodology to organize and cluster objects on the air channel. This work proposes a method that: (i) respects the ordering of objects represented by an object graph, (ii) attempts to cluster related objects close to one another, and, (iii) provides more availability of frequently requested objects.

The structure of this paper is as follows: Section 2 overviews the underlying environment. Section 3 discusses the related work that has been suggested for object clustering and organization on conventional storage (disk(s)) and air channel(s), respectively. The proposed algorithms for the air channel are introduced in Section 4. These algorithms are simulated and the results are analyzed in Section 5. Finally, Section 6 summarizes the work and lists future research directions.

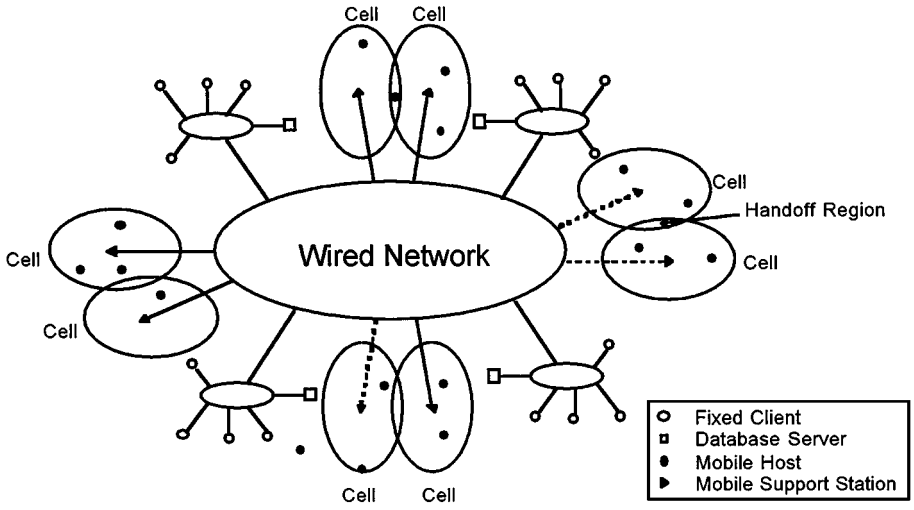


Figure 1. Mobile information system.

2. Environment

Broadcasting information is not a new concept. Whether through a guided (such as a cable) or unguided medium (such as air) the principle is the same. The concept is based on the encoding and transmission of the desired signal (analog or digital) on a certain frequency. The encoded information is supplied by one source and read by multiple receivers. Broadcasting has been used extensively in multiple disciplines.

Figure 1 depicts an overview of the environment being assumed. The data to be broadcast is available at the database server. The data is transmitted to the mobile support stations (MSSs). A MSS broadcasts the data on a certain air channel and is responsible for servicing the mobile hosts (MHs) within its cell. The MHs within the corresponding cell are capable of tuning to the air channel and retrieving such data. Since broadcasting provides the information only in one direction. The bandwidth does not need to be scaled as the number of users accessing the channel increases. Therefore, broadcasting is an attractive solution for the challenge of a limited bandwidth. Also, since the information on the air could be considered "storage on the air", it offers an elegant solution to compensate for the limited resources at the MH. Broadcasting is an ideal medium for applications that utilize public and location-dependent data—e.g., location of emergency vehicles, road tariffs, traffic maps, weather maps, stock prices and graphs, taxi dispatching, mail tracking, airline schedules, news, and locations of color printers in the vicinity.

We refer to the set of all broadcast data elements as the *broadcast*. A broadcast is performed in a cyclic manner. The MH can only read from the broadcast, whereas the database server is the only entity that can write to the broadcast. Since dependencies can exist among the data items within the broadcast, and since data items might be replicated within a broadcast, it is assumed that updates of data elements within a broadcast are only reflected at the following cycle. This constraint enforces the integrity among the data elements within the

broadcast. In order to ensure synchronization among the different MSS, we assume that every broadcast contains an index of its data. When a MH leaves the cell of a certain MSS, the pointers pointing to the data on the broadcast become stale. Therefore, as a MH crosses the boundary of a new cell, it downloads (or looks up) the pointers to the data that it had intended to retrieve from the previous broadcast.

3. Object organization

Proper organization of data objects as a means of reducing latency has been the subject of intensive research during the past four decades. Whether the physical storage medium is a flat memory or a multiple-plate disk structure, an appropriate data placement algorithm should attempt to detect data locality and cluster **related** data close to one another. The objects in an object-oriented paradigm are normally associated with one another through several semantic connections—inheritance, aggregation, or association. An object-clustering algorithm takes advantage of such relationships and attempts to map a complex object into a linear sequence of objects along these semantic links. In many practical applications, object relationships are not random but can be expressed as a hierarchy or a directed acyclic graph (dag) in which objects are represented as the vertexes (or nodes) and edges (or links) are the relationships among these objects. Typical operations performed on a dag are the navigation of its nodes along its links. It has been shown that such clustering can improve the response time by order of magnitude.

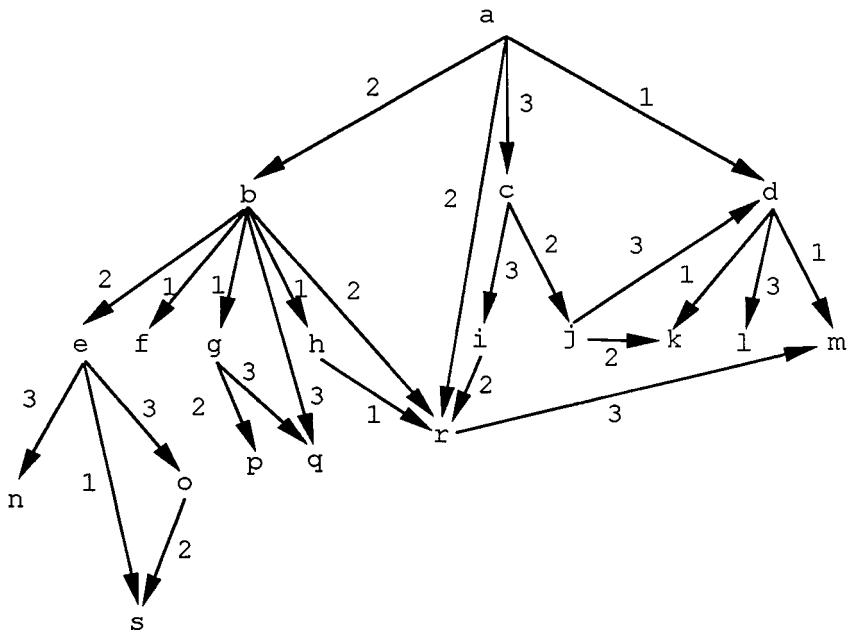
In the conventional computing environment, where objects are persistent data elements stored on disk(s), the clustering algorithms proposed in the literature are intended to place semantically connected objects physically along the sectors of the disk(s) close to one another. The employment of broadcasting in the mobile-computing environment motivates the need to study the proper data organization along the sequential air channel.

3.1. Disk(s)

Banerjee et al. [4] suggested three types of clustering sequences in a CAD object-oriented database-*depth first*, *breadth first*, and *children-depth-first*. Chang et al. [7] developed a clustering algorithm for objects with multiple relationships and different access frequencies among objects. Based on the notion provided by [7], Cheng et al. [10] analyzed the clustering technique by studying the effect of object updating when tuning the disk page *read/write* ratio.

Chehadeh et al. [9] investigated object clustering in a parallel disk environment; namely, clustered, synchronized, declustered, and declustered-synchronized disks. The work analyzed the conceptual changes required to map the clustering scheme proposed in [10] onto parallel disks. Finally, Lim et al. [16] studied the employment of the clustering scheme of [10] in a distributed environment.

Figure 2 depicts a weighted dag and the resulting clustering sequences achieved when different clustering techniques are applied. It should be noted that the techniques proposed in [4] are only applicable to non-weighted dags and that the techniques proposed in [7] and [10] generate the same clustering sequence.



Clustering method	Resulting sequence
Depth First [4]	abensofgpqhrmcijdkl
Breadth First [4]	abrcdefgqhijklmnsop
Children-Depth First [4]	abrcdefgqhmnsopijkl
Level Clustering [10]	acibgqprmenosjdlkfh

Figure 2. Graph and various clustering methods.

3.2. Air channel

A disk and an air channel have different mediums; in addition, they have major structural and functional differences. The disk has a three-dimensional structure made out of tracks on multiple plates composing cylinders (disks can have a four-dimensional structure if multiple disks are used). An air channel, on the other hand, is a one-dimensional structure (similar to that of the magnetic tape). The disk has a random-access feature and the air channel is sequential in nature. Information supplied by the air channel can not be backed up and re-read. Finally, the current raw data rate of a disk is generally much larger than that of the air channel.

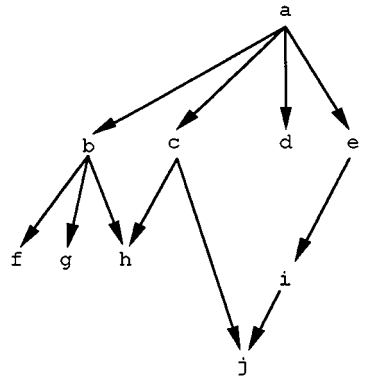
The works of Zdonik et al. [20] and Acharya et al. [1] investigated two issues, namely, the mapping of disk pages onto a broadcast channel and the effect that such organization might have on the management of cache at the MH. These works brings a new and creative idea of how disk pages should be mapped along the air channel. In that research, if a data

item is to be broadcast more frequently, then the entire page is broadcast more frequently. In addition, due to the plain structural nature of the page-based environment, the research looked at the pages as abstract entities and hence did not consider the page contents (data and its semantics) in generating the order of the pages. In object-oriented systems, directed semantics among objects greatly influence the method in which objects are retrieved, and thus, have their direct influence on the placement of these objects/pages.

4. Object organization on an air channel

In spite of the differences between a disk and an air channel, from a database perspective, the goal in organizing information on the air channel is still the same as that of the disk—to reduce the response time for accessing a series of objects requested. In order to achieve this goal, the object organization on an air channel has to meet the following three criteria:

1. **Linear ordering:** As noted before, the air channel is a single-dimension sequential access structure. This fact requires that the object ordering be linear (or topologically ordered). In a dag representing an object schema structure, an edge between two nodes could signify an access pattern among the two nodes; retrieving the object representing the first node could trigger the retrieval of the second object. Therefore, if an edge exists from an object `a` to an object `b`, then to achieve minimum delays between object(s) retrieval(s), a has to appear before b. We define the *linearity* property to be the fact that if an edge exists between two objects o_1 and o_2 and in the direction $o_1 \rightarrow o_2$, then o_1 should be placed prior to o_2. As can be seen in figure 2, the solutions provided for object(s) clustering on a disk do not satisfy the linearity property. This is not a drawback in the algorithms, but rather a feature that was not required due to the multidimensional structure and random access capability of the disks.
2. **Minimum linear distance between related objects:** In a query, multiple objects might be retrieved following their connection pattern. Reducing the distance among these objects along the broadcast reduces the average delay generated while waiting between the retrieval of two objects. This requirement is also valid within the disk environment and supported by the disk-based clustering algorithms.
3. **More availability for popular objects:** In a database, not all objects are accessed with the same frequency. Generally, requests to data follow the 20/80 rule—a popular small set of the data (20%) is accessed the majority of the time (80%). Considering the sequential access pattern of the broadcast channel, providing more availability for popular objects can be achieved by simply replicating such objects. One mechanism to determine which objects are more popular than others utilizes a feedback channel. Query engines at the MHs can report back, via the feedback channel, the object identifiers that were retrieved. Statistical analysis can be applied to such data in order to study the access patterns and objects that are more popular than others. Utilizing feedback channels assumes a two-way communication structure. This fact increases the traffic on the air channels. However, such increase is relatively minor, since the information provided from the MH to the MSS is of statistical type nature (e.g., a list of the objects identifiers retrieved during this broadcast).



#	Linear	Individual costs											Total
	sequence	ab	ac	ad	ae	bf	bg	bh	ch	cj	ei	ij	cost
1	abfgchdeij	1	4	6	7	1	2	4	1	5	1	1	33
2	abfgcheijd	1	4	9	6	1	2	4	1	4	1	1	34
3	abcdefghij	1	2	3	4	4	5	6	5	7	4	1	42
4	abgfeichjd	1	6	9	4	2	1	6	1	2	1	3	36
5	acdeijbhgf	6	1	2	3	3	2	1	6	4	1	1	30
6	adeicjbhgf	6	4	1	2	3	2	1	3	1	1	2	26
7	adecbihgfj	4	3	1	2	4	3	2	3	6	3	4	35
8	adecbhgfij	4	3	1	2	3	2	1	2	6	6	1	31
9	adecijbhgf	6	3	1	2	3	2	1	4	2	2	1	27
10	adbfgcheij	2	5	1	7	1	2	4	1	4	1	1	29
11	adceijbhgf	6	2	1	3	3	2	1	5	3	1	1	28
12	aeidcjbhgf	6	4	3	1	3	2	1	3	1	1	3	28
13	aedcbihgfj	4	3	2	1	4	3	2	3	6	4	4	36
14	aedcijbhgf	6	3	2	1	3	2	1	4	2	3	1	28

Figure 3. Graph, linear sequences, and costs.

4.1. Linearity and minimum linear distance

Figure 3 depicts a directed graph and multiple linear sequences that satisfy the linear property. The individual middle columns represent the cost of delays between every two objects connected via an edge. For the sake of simplicity and without loss of generality, we use an *object* unit as a unit of measurement and assume that all objects are of equal size. The cost associated with an edge between a pair of objects is calculated by counting the number of objects that separate these two objects in the linear sequence (including the starting object). For example, in the first row, for the link *ad*, objects *a* and *d* are separated by the sequence *bfgch* and thus have a cost of 6. Note that ideally, we would like every edge to have a cost of 1. However, due to the linear sequence, such a requirement is impossible to achieve. The right-most column represents the total cost associated with each individual linear sequence. An optimal sequence is the linear sequence with the minimum total sum. In a query where multiple related objects are retrieved, a minimum average linear distance translates into minimum average delay between successive object retrievals, and

thus, smaller overall response times. In this example, the optimum linear sequence achieves a total sum of 26 and is shown in bold borders in the sixth row.

One method in obtaining an optimal linear sequence is to enumerate and calculate all possible linear sequences with their associated costs and then choose the sequence with the minimum cost. Naturally, such a solution, though simple, is computationally impractical. We propose a set of heuristic rules that generate a linear sequence with a reasonable cost.

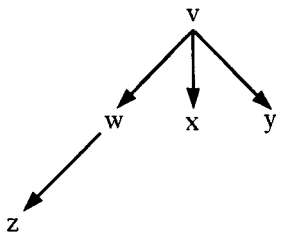
Definition 1. A *free* node is a node that has either one or no parent. In figure 3, node *e* is a free node whereas node *h* is not. A graph containing only free nodes makes up a forest.

Heuristic rules.

(i) Order the children of a node based on their number of descendants in ascending order—the child with the least number of descendants is placed first in the sequence.
(ii) Once a node is selected, all its descendants should be visited and placed on the sequence in a depth first manner, without any interruption from breadth siblings.
(iii) If a node has a non-free child with all of its parents already visited, the non-free child should be inserted in the linear sequence before any free child.

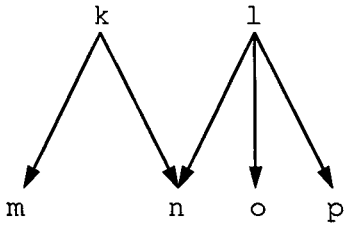
The advantage of the first two heuristics can be seen in figure 4. The most efficient sequence is the fourth one, where the nodes are selected based on the number of descendants in an ascending order (x, y, or (y, x) then w) and depth-first descendants (z) are not interrupted by breadth siblings (x and y). The other possible sequences do not comply with the heuristics and thus incur higher overall costs.

The advantage of the third heuristics is depicted in figure 5 (which shows a graph with a non-free node (n)). The third sequence complies with the third heuristic (along with the



Linear Sequenc	Individual vw	vx	vv	wz	Total Cost
vwzxv	1	3	4	1	9
vwxvz	1	2	3	3	9
vxwvz	2	1	3	2	8
vxvwz	3	1	2	1	7

Figure 4. Rooted all free-node graph (tree).



Linear Sequenc	Individual Costs					Tota Cost
	k	kn	ln	lo	lp	
kmlpon	1	5	3	2	1	12
kmlonp	1	4	2	1	3	11
kmlnop	1	3	1	2	3	10

Figure 5. Graph with non-free node.

first two) by placing *n* right after inserting *l* in the sequence. And thus resulting in the least overall cost.

Algorithm APPROXIMATELINEARORDER implements these heuristics and summarizes the sequence of the operations required in obtaining a linear sequence. The algorithm assumes a greedy strategy and starts by selecting a node with an in-degree of zero and out degree of at least one. The algorithms operates by satisfying the three heuristics. The DFS assumption made in lines 1 satisfies the second heuristic. The first if statement guarantees the third heuristic, and finally the node selection process in line 9 satisfies the third heuristic.

APPROXIMATELINEARORDER

```
1  traverse dag using DFS traversal
2     append the traversed node N to sequence
3     remove N from {nodes to be traversed}
4  if {non-free children of N having all their parents already in the sequence} ≠ ∅
5     Set ← {non-free children of N having all their parents already in the sequence}
6  else
7        if {free children of N} ≠ ∅
8        Set ← {free children of N}
9  NextNode ← node ∈ Set | node has least # of descendants among the nodes in Set.
```

Applying this algorithm to the graph of figure 3 generates either the fifth or eleventh sequence. This is dependent on whether *c* or *d* was chosen first as the child with the least number of free-children. As one can observe, neither of these sequences is the optimal sequence; however, they are reasonably better than other sequences and can practically be obtained in polynomial time.

It should be noted that nodes not connected to any other nodes—nodes with in-degree and out-degree of zeros—are considered harmful and thus are not be handled by the algorithm. Having them in the middle of the sequence introduces delays between objects along the

sequence. Therefore, we exclude them from the set of nodes to be traversed and handle them by appending them to the end of the sequence. In addition, when multiple dags are to be mapped along the air channel, the mapping should be done with no interleaving between the nodes of the dags. The interleaving should not be performed, since it would introduce additional nodes in between the original nodes of the individual dags. When mapped along the air channel, the interleaving translates into additional delays between the objects.

4.2. Varying levels of connectivity

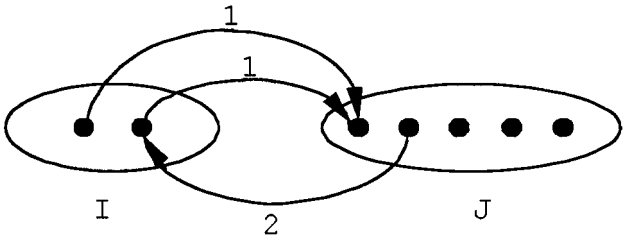
In a complex object, objects are connected through semantic links. Moreover, in practice, the degree of connectivity among objects varies. The frequency of accesses of objects in an object-oriented database reveals that some patterns are more frequently traversed than others [7, 10]. This observation motivates us to develop a new algorithm that attempts to cluster strongly connected objects closer to each other. The algorithm PARTIALLYLINEARORDER takes a weighted dag as its input and produces a linear sequence. It combines the nodes (single_nodes) of the graph into multi_nodes in a descending order of their connectivity (semantic links). The fist insertion of single_nodes within a multi_node has to respect the linear order at the granularity level of the single_nodes. The multi_nodes are merged (with multi_nodes or single_nodes) at the multi_node granularity, without interfering with internal ordering sequences of a multi_node. In the algorithm, the for loop in lines 2–3 merges single_nodes together to generate multi_nodes. The for loop starting at line 4 merges the multi_nodes with adjacent multinodes or single_nodes (note that *AN* could be either a multi_node or a single_node). To guarantee an average minimum distance among related objects, the ordering of a merge between a multi_node and an adjacent node is based on the shorter weighted linear distances between the two of them in both directions. The following formula is used to measure the weighted linear distance:

$$\textit{Weighted Linear Distance}_{IJ} = \sum_{\textit{All Edges } I \rightarrow J} \frac{w_{ij}}{\textit{length}[\textit{multi\_node}_I] - \textit{order}[\textit{single\_node}_I] + \textit{order}[\textit{single\_node}_J]}, \tag{1}$$

where *length*[*multi_node*] is the number of single_nodes within a multi_node and *order*[*single_node*] is the location number of the single_node within the multi_node. Figure 6 shows an example of merging two multi_nodes and the associated distances in both direction. As can be seen, although one multi_node is connected to the other via a higher weight (2), the other is connected via a smaller weight (1) twice. Therefore, in this example the exact order would be I followed by J. Figure 7 depicts an example of the running process of this algorithm.

PARIALLYLINEARORDER

1 **for** every weight w_s in descending order
2 **for** every two nodes N_1 & N_2 connected by w_s
3 merge N_i & N_j into one multi_node



$$\textit{Weighted Linear Distance}_{IJ} = \frac{1}{2-1+1} + \frac{1}{2-2+1} = 1.5$$

$$\textit{Weighted Linear Distance}_{JI} = \frac{2}{5-2+2} = 0.4$$

Figure 6. Linear distance in multi-nodes.

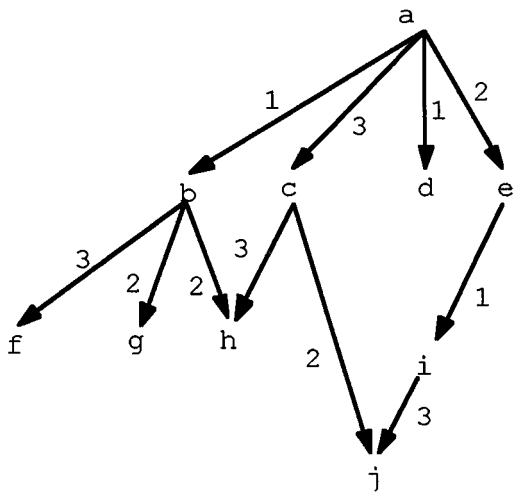
4 **for** every multi_node *MN*
5 $w_m = w_s - 1$
6 **for** every weight w_m in descending order
7 **while** ∃ adjacent_node *AN* connected to *MN*
8 **if** ∃ an edge in both directions between *MN* & *AN*
9 compute $\textit{WeightedLinearDistance}_{MN\_AN}$ & $\textit{WeightedLinearDistance}_{AN\_MN}$
10 merge *MN* & *AN* into one multi_node, based on the appropriate direction

4.3. Object replication

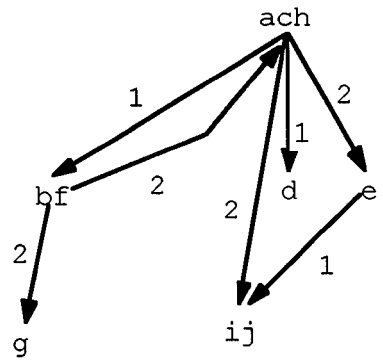
An appropriate mapping scheme should allow the replication of some popular objects on the air channel. Such a scheme should take three main issues into consideration:

(i) The number of times an object should be replicated,
(ii) The region(s) within the broadcast where an object should be replicated, and
(iii) The distribution of replicated objects in each region.

1. **Number of replications.** We assume that there is a feedback channel through which the server is capable of measuring the retrieval frequency of each object. The information on these feedback channels is directed from the MHs to the MSS. Such information could be represented as a string of object identifiers (OIDs) accessed by the MHs. The string can be divided into two parts. One part would include the OIDs of objects accessed through their semantic links, and the other would include OIDs of objects accessed on a random basis. The popularity of each object is proportional to the retrieval frequency. Hence, the retrieval frequency is used as a means for determining the degree of replication for each object. It should be noted that assuming a feedback channel results in increasing the traffic (and hence reducing the bandwidth per user) on the air channels. However,



(a)



(b)

bfgacheijd

(c)

Figure 7. Process of PARTIALLYLINEARORDER. (a) Original graph, (b) First and second iterations, and (c) Third iterations.

this increase is only minimal, since it only includes statistical information of the objects that were accessed by the MSS and does not include lengthy information such as queries or response to queries.

2. **Replication region.** Both APPROXIMATELINEARORDER and PARTIALLYLINEAR-ORDER algorithms attempt to map a dag into a linear sequence. The appearance of each object on this linear sequence (*MainCopy*) partitions the air channel into two regions—

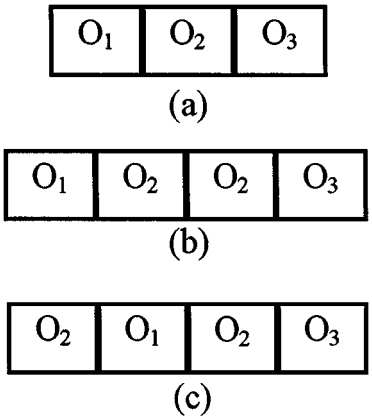


Figure 8. Example of replicated object distribution.

the region before and after the *MainCopy*. An object can be accessed either randomly or via its semantic link. In order to preserve the connections from ancestors leading to this object, a popular object that is generally accessed via its semantic links should be replicated after the *MainCopy*. Therefore, for each object, the number of random accesses and the number of references via the semantic links (obtained via the feedback channel) are used to determine the number of replications to be distributed along the entire broadcast and those to be distributed strictly following the *MainCopy*, respectively.

3. **Replication distribution.** Although the main advantage of replicating popular objects is to reduce the amount of delays for these objects, the incurred disadvantage is the extra delays introduced for the non-replicated objects [1]. To show this, consider figure 8. Originally three objects O_1, O_2, and O_3 are available on the broadcast (a). Assume that the probabilities for accessing these three objects are the same (each at 0.33). Assume that the access on average hits the broadcast at the mid point of an object, all objects are equal in size, and the broadcast in successive cycles have the same objects in the same order. The average delay for every object (in object units) would be:

$$0.33[0.33(2.5) + 0.33(1.5) + 0.33(0.5)] = 0.5 \text{ objects}$$

(The value outside the square bracket is the probability and the value within the square bracket is the average distance to be traveled). Taking the sum of the three equal average delays for the three objects, we get an overall average delay of 1.5 objects.

Assume that O_2 is a popular object and should be replicated. Figure 8(b) shows one possible layout for the four objects. Let us consider the average delays for the objects in (b). For objects O_1 and O_3 it is:

$$0.33[0.25(3.5) + 0.25(2.5) + 0.25(1.5) + 0.25(0.5)] = 0.67 \text{ object}$$

whereas for object O_2 it is:

$$0.33[0.25(0.5) + 0.25(0.5) + 0.25(2.5) + 0.25(1.5)] = 0.4167 \text{ object}$$

Taking the sum of the four average delays for the three objects, we get an overall average delay of:

$$0.67 + 0.67 + 0.4167 + 0.4167 = 1.75 \text{ object}$$

This analysis suggests that the overall average delay increased when replication is used. However, this result is misleading due to two reasons: First, the analysis assumed that all object had equal probabilities of being accessed. Such an assumption contradicts the fact that popular objects have higher probability of being accessed than no-popular ones. Second, the distribution used in placing both copies of O_2 in figure 8(b) did not distribute the copies uniformly. This fact does not agree with the basic assumption that the probability of accessing the broadcast at any time is uniform. Observing these two points, we see the need to conduct the analysis using a **uniform** distribution, while varying the probability of accessing a popular (replicated) object. A uniform distribution is shown in figure 8(c). Figure 9 plots the resulting overall average delay vs. the object access

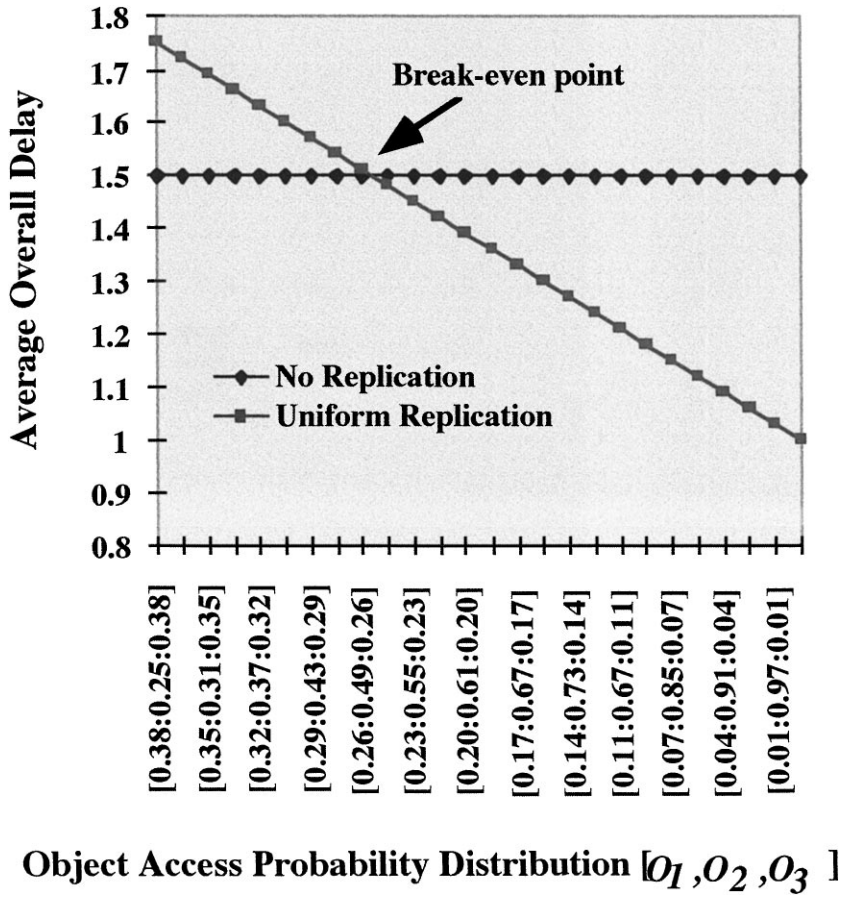


Figure 9. Replication break-even point.

probability distribution for both the no-replication and the uniform replication figure 8(c) cases. The sum of the distribution of the probabilities (given in terms of $[O_1, O_2, O_3]$) at any time for is equal to 1. As can be seen from the figure, the average delay for the no-replication case stays the same throughout. This is due to the fact that although the average delay for the individual objects changes, the average distance to be traveled is the same for all objects. Therefore, the overall sum ends up to be the same. For the replication case, the average distance to be traveled is shorter for O_2 than for O_1 and O_3. Therefore, as the probability of access for O_2 increases, the overall average delay decreases. We refer to the intersection point of the two curves as the replication-break-even point (*RBEV*).

Definition 2. *RBEV* is a point beyond which replication proves to be profitable.

In this example, for an object access probability distribution larger than [0.25, 0.5, 0.25], replication should be used for O_2.

Algorithm REPLICATION summarizes the sequence of the operations necessary to perform in obtaining the entire broadcast (including all the replicated objects in their appropriate locations) at the beginning of every broadcast cycle. The following symbols are used in the algorithm:

SetObjToBeRep	The set of all popular objects
NumObj	Number of objects on the broadcast excluding any replicated copies
RandomRepCopies[*o*]	Number of copies of object *o* to be replicated due to random access to object *o*
ConenctionRepCopies[*o*]	Number of copies of object *o* to be replicated due to accessing *o* based on a connection
RepFrequency[*o*]	Number of copies object *o* is to be replicated as a function of the interest of users. It is equal to *RandomRepCopies*[*o*] plus *ConenctionRepCopies*[*o*]
AccessProb	Probability of accessing a certain object from *SetObjToBeRep*
ObjAccessProbDist[*o*]	Access probability distribution of *o* (point on the x-axis)

As can be seen from the algorithm, the first for loop calculates the maximum number of objects to appear on the broadcast. The second for loop selects the objects to be replicated in an ascending order of their access probability. In order to obtain the *RBEV*, both the no replication and the uniform replication data (the curves in figure 9) have to be obtained. Note that the no replication data is the same for all objects. If the required access probability distribution has a value along the x axis that is less than the x-axis value of the *RBEV*, then the replication of **all** objects (including the object being considered) with higher access probabilities would be beneficial and should be included on the broadcast. Otherwise, the object being considered should not be replicated and the process is repeated for object with the next higher access probability. The final for loop uniformly replicates the desired object either after the *MainCopy* or throughout the entire broadcast.

REPLICATION

```
1   Generate linear sequence
2   for each o ∈ SetObjToBeRep
3     TotalNumObj ← NumObj + RepFrequency[o]
4   ObjToBeRep ← Number of objects in SetObjToBeRep
5   for each o ∈ SetObjToBeRep in ascending order of AccessProb
6     Obtain RBEV
7     if ObjAccessProbDist[o] < RBEV
8         TotalNumOfObj ← TotalNumObj − RepFrequency[o]
9         ObjToBeRep–
10      else
11        break
12  for j ← ObjToBeRep downto 1
13    o ∈ SetObjToBeRep in descending order of AccessProb
14    Distribute RandomRepCopies[o] uniformly along broadcast
15    Distribute ConenctionRepCopies[o] uniformly after MainCopy
```

5. Performance evaluation and results

A simulator was developed to study the behavior of the proposed mapping algorithms based on a set of rich statistical parameters. In addition, as part of our study, we also took steps to develop a testbed to measure the effectiveness of these algorithms. Our testbed is an object-oriented database. The OO7 benchmark [6] (due to its rich schema graph for a general object-oriented database application) was chosen to generate the access pattern graphs. The drawback of the OO7 benchmark is that it assumes a CAD application and thus provides meta data accordingly. In a broadcasting environment, the applications are normally assumed to have "public" data which is to be shared among a large population of users. Therefore, we choose a financial database as our underlying database. We used the NASDAQ exchange as our base model, where data is in both textual and multimedia (graphics—i.e., graphs and tables) formats.

5.1. Parameters

Table 1 provides a brief description of the input and output parameters. The simulator is designed to measure the average access delay for the various input parameters. Table 2 provides a listing of the input parameters, along with their default values and possible ranges. The default values are set as the value of the parameter when other parameters are varied during the course of the simulation. The ranges are used when the parameter itself is varied.

- **Number of nodes:** This parameter represents the number of objects to be mapped along the air channel. Since the number of securities within the NASDAQ exchange is 5464 [17], we chose 5000 as our default value. Note that only 10% of the securities mostly

Table 1. Description of parameters.

Parameter	Brief description
	Input parameters
Number of nodes	Number of objects within the graph (excluding replication)
Object size	Sizes of objects (small/medium/large)
Object-size distribution	Distribution of the sizes of objects within the database
Next-node ratio	Connectivity to next node (through a random request or via a connection)
Out-degree distribution	Distribution of the type of nodes based on their out-degrees
Level distribution	Semantic connectivity among two objects (weak/normal/strong)
Percentage of popular objects	Percentage of objects requested more often than others
Replication frequency	The number of times a popular object is to be replicated
	Output parameter
Average access delay	In a single query, the average delay between accessing two objects

Table 2. Input-parameter values.

Parameter	Default value	Ranges
Number of nodes	5000	400–8000
Object size (in bytes)		
Small	$2 \leq 0 < 20$	2–20
Medium	$20 \leq 0 < 7$ K	20–7 K
Large	7 K $\leq 0 < 50$ K	7 K–50 K
Object-size distribution [$S : M : L$]	1 : 1 : 1	0–6 : 0–6 : 0–6
Next-node ratio [$C : R$]	8 : 2	0–10 : 10–0
Out-degree distribution [0 : 1 : 2 : 3]	3 : 3 : 2 : 1	1–6 : 1–6 : 1–6 : 1–6
Level distribution [$W : N : S$]	1 : 1 : 1	1–4 : 1–4 : 1–4
Percentage of popular objects	20%	10–50%
Replication frequency	2	1–10

are active on a daily basis. To achieve more insight about the behavior of our algorithms, this parameter was varied in the range of 400 to 8000.

- **Object-size and size distribution:** In a database, objects are of variable sizes. The objects within our assumed database could be as simple as a security quote or as complicated as a complex object representing a collection of simpler objects—graphs and figures. In our study, the object size is classified as small, medium, and large. A small object is assumed to be less than 20 bytes long. In order to gain more insight on the size(s) of objects to be represented in a graphic format, we conducted a simple experiment by plotting a security graph in different graphic formats. The graph had the physical dimension of 4×3 inches (358 pix) $\times$ (481 pix). Such dimensions are reasonably within the size of the displays of the majority of portable units. The results of the experiment are shown

Table 3. Object sizes for various graphics formats.

Format	Resolution	No. of colors	Size (in bytes)	
			Regular	Compressed
Gif	480×360	16	6977	Same
	1024×768	256	421189	Same
Sun raster	480×360	16	172609	7519
	1024×768	256	789079	613057

in Table 3. Based on these results, we choose 7 and 50 K bytes as the maximum size(s) for medium and large objects, respectively. In the simulation, an object size is selected on a random basis within its range. A distribution $S : M : L$ of $1 : 1 : 1$ means that the database is composed of an equivalent number of small, medium and large objects. As the number of objects is fixed, ranging the distribution of the object sizes translates into mapping less or more data along the air channel.

- **Next-node ratio:** During the course of a query, objects are either accessed via the existing semantic links among them or in a random fashion. This parameter defines the ratio between accessing the next object, either along a semantic link or on a random basis. A C : R ratio of $10 : 1$ means that for all accesses between two objects, 10 are based on a connection basis and 1 on a random basis. For our simulation, and based on the OO7 benchmark, we choose a default ratio of $8 : 2$.
- **Out-degree distribution:** This parameter indicates the distribution of the number of out edges stemming from a node within the graph. An out-degree of 0 indicates a sink node. Analyzing the connections within the class hierarchy of the OO7 benchmark, we chose a ratio of distribution of out-degrees of 0, 1, 2, and 3 of $[3 : 3 : 2 : 1]$ as the default value. We ranged the value of the distribution of each out-degree between 1–6.
- **Level distribution:** The level of connectivity between two objects represents the semantic distance between the two objects. Based on [7, 9, 10], we distinguished nodes with three levels of connectivity–weak, normal, and strong. A level distribution of $[1 : 1 : 1]$ means that the connectivity among the objects within the database is equally divided into weak, normal, and strong connections. We choose a distribution of $[1 : 1 : 1]$ as the default value and varied the range for each level between 1 and 3.
- **Percentage of popular objects and replication frequency:** Considering our financial database, it is natural to assume that a security that has more investors is accessed more often than the others. Due to the 20/80 rule, we set 20% as the default value for the percentage of popular objects in the database. The replication frequency designates the frequency at which popular objects are to be replicated on the air channel. We choose 2 as the default value for the frequency of replicating all popular objects and vary this number between 1 and 10.
- **Average access delay:** The output parameter is used to measure the effectiveness of the proposed organization scheme. This work is intended to reduce the delay(s) incurred in accessing a series of objects. We define the access delay as the time elapsed between the end of accessing one object and the start of accessing another within the same query.

5.2. Simulation and results

The simulator is composed of two stages. The first entails structuring the access pattern object graph, based on certain statistical parameters, and mapping it along the air channel using various mapping algorithms. In order to get a wide spectrum of possible graphs, we define a set of parameters that have their influence on how both algorithms map the objects along the channel. These parameters are: (i) the percentage of non-free nodes, (ii) the depths of all trees that exist within the graph, and (iii) the amount of sharing that exist between trees through non-free nodes. Varying these statistical parameters, we generated 500 access graphs that were used as part of our testbed. In addition, we simulated three mapping algorithms. The first was a non-linear children-depth first clustering algorithm [4]. This clustering scheme was chosen, since it outperforms the conventional schemes for non-weighted dags in the conventional environment. The second and third were based on our suggested schemes using PARIALLYLINEARORDER and APPROXIMATELINEARORDER algorithms.

In the second stage, our simulator generated requests and accessed the required objects from the air channel. Similar to the queries in the OO7 benchmark, during each run, in each query an average of 20 objects were accessed. These objects were accessed either through their semantic links or randomly (following the $[C : R]$ value of the next-node ratio). The simulator measured the average access delay. In order to generate each point, the simulator was run 100,000 times. We assumed a broadcast data rate of 1M bit/sec and show the results in terms of seconds.

- **Number of objects:** Figure 10 shows the effects of varying the number of objects on the average access delay. Both proposed schemes (APPROXIMATELINEARORDER and

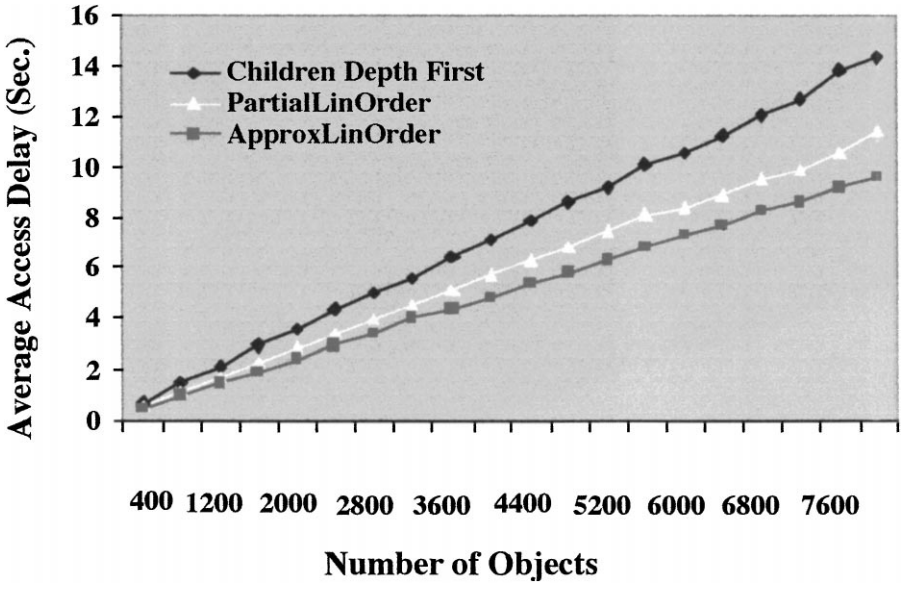


Figure 10. Total number of objects.

PARTIALLYLINEARORDER) performed better than the conventional children-depth first. This is due to the fact that the linearity feature is not supported in the children-depth first algorithm. As can be seen in all three cases, the average access delay increased as the total number of objects increased. The mapping of additional nodes on the broadcast introduced extra delays between the retrieval of two consecutive objects. Taking a closer look at this effect, we observe that this extra delay is mainly due to an increase in the distance for objects that are retrieved randomly (not based on their semantic links). Since the goal of both algorithms is to cluster semantically related objects close to one another, adding extra objects had only a minimal effect on the distance separating semantically related objects. The APPROXIMATELINEARORDER algorithm outperformed the PARTIALLYLINEARORDER algorithm. This is due to the fact that the latter attempts to cluster strongly connected objects closer to one another than loosely connected ones and, hence, compromises the linearity property for the loosely connected objects. This compromise overshadowed the benefit and is amplified as the number of objects increased.

In order to get a better insight on how the results of our proposed schemes compare with the optimal case, we constructed two graphs each with 10 nodes. We excluded any replication from this experiment. The default values of all the input parameters were used (except for the replication parameters). We generated the optimal sequence through an exhaustive search among all the possible sequences, and ran both of our proposed schemes. The average access delay was measured. The results of APPROXIMATELINEARORDER and PARTIALLYLINEARORDER were 79 and 76%, respectively, of the values achieved by the optimal case.

- **Size distribution:** Figure 11 shows the effects of varying the distribution of the object size among small, medium and large. As expected, the smallest average access delay took place when the air channel contained smaller objects (point $[6:0:0]$). However, as the population of objects shifted towards the larger objects, the average access delay increased (point $[0:0:6]$).
- **Next-node Ratio:** Figure 12 depicts the effect of varying the ratio of the next-node access type. At one extreme ($C:R=10:0$), all objects are accessed along the semantic links. In this case, since all connected objects are clustered close to one another, the average access delay is at a minimum. The delay, however, increased as more objects are accessed randomly. Also, where all the accesses are on a random basis, clustering (and linearity) does not improve the performance, and all mapping algorithms perform equally.
- **Out-degree distribution:** Figure 13 shows the effect of varying the out-degree distribution within the graph structure. The first point in the graph ($[9:0:0:0]$) assumes that all the nodes within the graph have an out-degree of zero—there is no semantic link among the objects. This is similar to stating that any access to any object within the graph is done on a random basis. In general, the average access delay is reduced as more connectivity is injected in the access graph. It is interesting to note that in our environment, it would be more desirable to deal with more, but simpler, objects than with few complex objects on the air channel. This is mainly due to the fact that nodes with higher out-degrees change the structure of the graph causing it to become wider rather

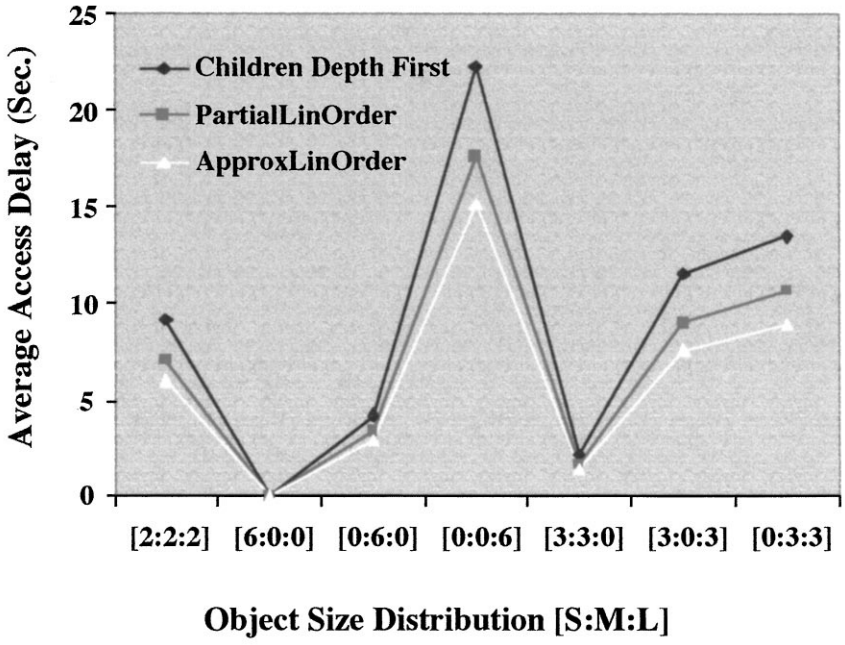


Figure 11. Object size distribution.

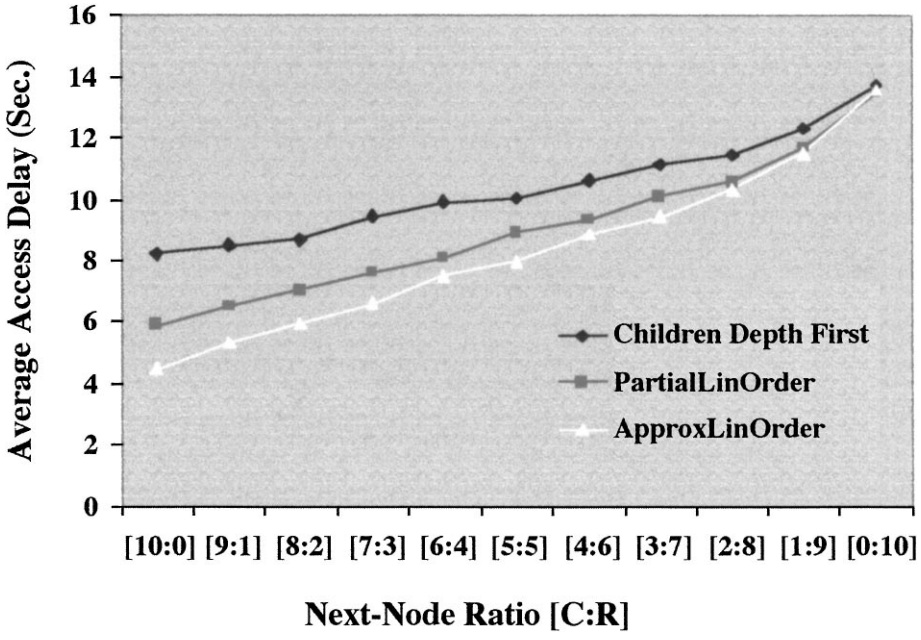


Figure 12. Next-node ratio.

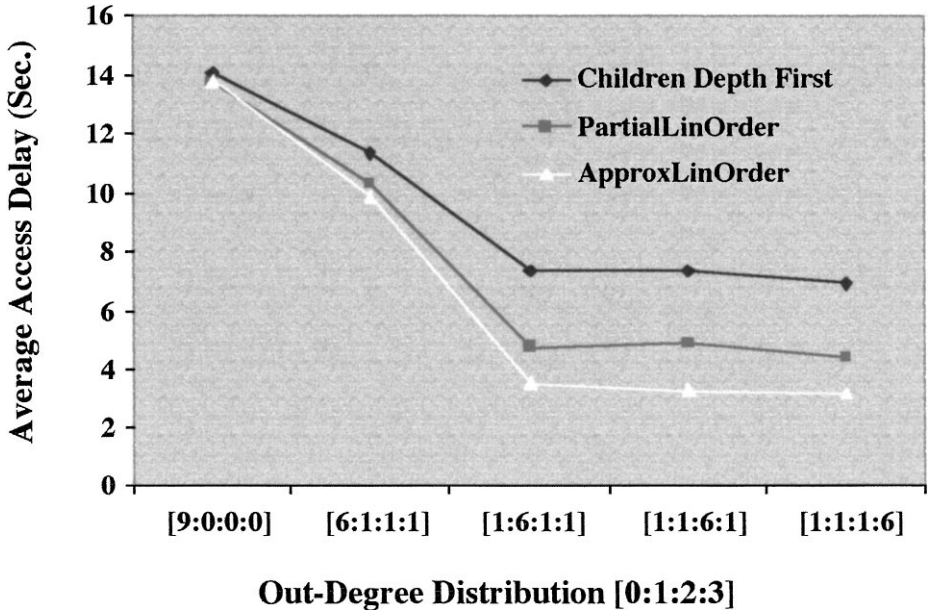


Figure 13. Out-degree distribution.

than deeper, and thus introducing more "interrupting" breadth nodes which the suggested algorithms attempts to avoid.

- **Percentage of popular objects and replication frequency:** The simulator also measured the effect of varying the percentage of popular objects and the replication frequency. The effect of these two parameters on the total number of objects on the air channel is identical. This is because increasing the percentage of popular objects replicates a higher percentage of the original objects along the air channel, and increasing the replication frequency replicates the same popular objects more frequently on the air channel. Therefore, from the access pattern perspective, the semantic of the accesses are different however, their effects on the average access delay is the same. In both cases, the average access delay increased as either parameter increased. Figure 14 shows the effect of modifying the percentage of popular objects.
- **Level distribution:** Throughout the previous analyses, we observed that the approximation algorithm outperformed, in general, the optimal partially non-linear one (the justification for that was provided when the effect of increasing the total number of objects in the system was discussed). In this section, we take a closer look at the details of the mapping achieved by the PARTIALLYLINEARORDER. Setting all our input parameters at their default values, we look at the average access delay for the objects connected at different levels. We observe that the average access delay for objects connected through strong connections is about 4.3 sec, whereas it is 7.3 and 7.6 sec for normal and weak connected objects, respectively. (Note that it is the weighted average of these three values along with that related to the randomly-accessed objects that produces the overall average shown in all the previous graphs.) The interesting point in obtaining these

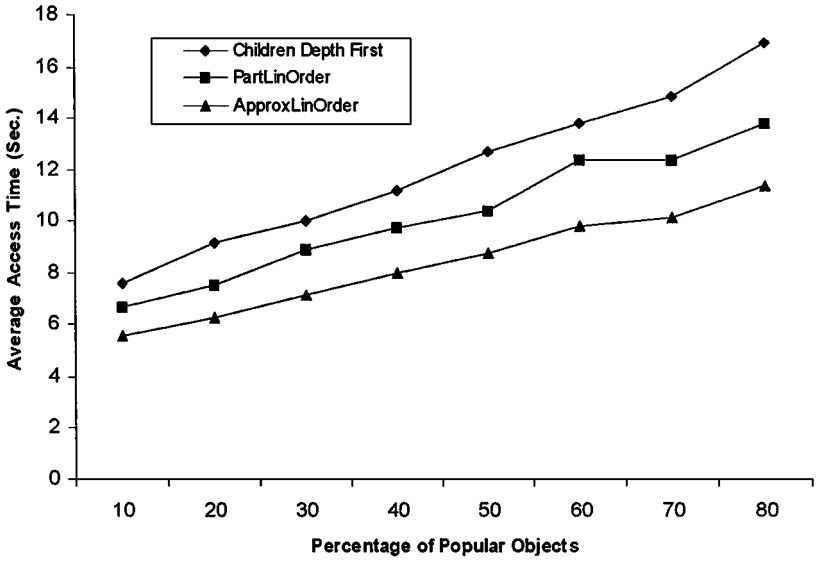


Figure 14. percentage of popular objects.

results is that the improvement is considerable for the objects connected by a strong connection; however, for a normal connection the performance was close to that of the weak-connection case. This is due to the fact that the algorithm performs its best optimization for the strong-connection case. It does that by ignoring all weaker links and by clustering at the granularity of a single node. When multi_nodes are merged based on the weaker links, the merging is performed at the granularity of a multi_node. It is for this reason that for a general case, where a dag assumes all the default values, the PARTIALLYLINEARORDER algorithm performed worse than APPROXIMATELINEARORDER.

6. Conclusion and future directions

Several mapping algorithms to organize objects on the air channel were studied. Heuristics were used to perform the mapping in polynomial time. The proposed algorithms were intended to satisfy three requirements; namely, linearity, closer distance among strongly connected objects, and replication of popular objects. In order to meet the first two requirements, two mapping algorithms were provided. The APPROXIMATELINEARORDER algorithm is a greedy-based approximation algorithm that guarantees the linearity property and provides a solution in polynomial time. The PARIALLYLINEARORDER algorithm guarantees the linearity property for the strongest related objects and relaxes the linearity requirement for objects connected through looser links. An algorithm for objects' replication was also presented. The simulation results showed that the APPROXIMATELINEARORDER algorithm offers a slightly better performance than the PARIALLYLINEARORDER mapping algorithm at the expense of more complexity. However, both algorithms performed better than a non-linear algorithm.

This work attempts to find a mapping algorithm that reduces the access latency among retrieved objects. Although minimizing the delay (minimizing the overall response time) is essential in a database environment, for a mobile environment, other parameters such as the overall energy consumption during the course of a query should also be taken into the consideration [14]. We are currently investigating an indexing scheme for object retrieval from the broadcast channel that reduces energy consumption. Furthermore, due to the fact that multimedia objects are relatively larger than objects in many other applications, research should be conducted in identifying the proper buffering schemes that should be used. Conventional buffering schemes may assume a much larger storage requirement than what is available on the mobile unit and require a large amount of I/O activity, which is not inline with the power conservation policy required at the mobile unit.

Finally, this work concentrated on the proper mapping of database objects on a single air channel. In the disk environment, we analyzed the mapping of objects on multiple parallel disks [9]. Similarly, we are investigating the proper mapping of objects on multiple channels.

References

1. S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks: Data management for asymmetric communication environments," Proceedings ACM SIGMOD International Conference on the Management of Data, 1995, pp. 199–210.
2. R. Alonso and H.F. Korth, "Database system issues in nomadic computing," Proceedings ACM SIGMOD Conference on Management of Data, 1993, pp. 388–392.
3. M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik, "The object-oriented database system manifesto," Proceedings Conference on Deductive and Object-Oriented Databases, 1989, pp. 40–57.
4. J. Banerjee, W. Kim, S.-j. Kim, and J.F. Garza, "Clustering a DAG for CAD databases," IEEE Transactions on Software Engineering, Vol. 14, No. 11, pp. 1684–1699, 1988.
5. M.W. Bright, A.R. Hurson, and S.H. Pakzad, "A taxonomy and current issues in multidatabase systems," EEE Computer, Vol. 25, No. 3, pp. 50–60, 1992.
6. M.J. Carey, D.J. DeWitt, and J.F. Naughton, The OO7 Benchmark, CS Technical Report, University of Wisconsin, Madison, Jan. 1994.
7. E.E. Chang and R.H. Katz, "Exploiting inheritance and structure semantics for effective clustering and buffering in an object-oriented DBMS," Proceedings ACM SIGMOD Conference on Management of Data, June 1989, pp. 348–357.
8. Y.C. Chehadeh, A.R. Hurson, and M. Kavehrad, "Multidatabases in the mobile-computing environment: Issues and solutions," Technical Report CSE-97-016, Computer Science and Engineering Department, Penn State University, Sep. 1997.
9. Y.C. Chehadeh, A.R. Hurson, L.L. Miller, S. Pakzad, and B.N Jamoussi, "Application of parallel disks for efficient handling of object-oriented databases," Proceedings Fifth IEEE Symposium on Parallel and Distributed Processing, Dec. 1993, pp. 184–191.
10. J.-B.R. Cheng and A.R. Hurson, "Effective clustering of complex objects in object-oriented databases," Proceedings ACM SIGMOD Conference on Management of Data, 1991, pp. 22–27.
11. I. Chlamtac and Y.-B. Lin, "Mobile computing: When mobility meets computation," IEEE Transactions on Computers, Vol. 46, No. 3, pp. 257–259, 1997.
12. G.H. Forman and J. Zahorjan, "The challenges of mobile computing," IEEE Computer, Vol. 27, No. 4, pp. 38–47, 1994.
13. A.R. Hurson, S. Pakzad, and J.-B.R. Cheng, "Object-oriented database management systems," IEEE Computer, Vol. 26, No. 2, pp. 48–60, 1993.
14. T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Data on air: Organization and access," IEEE Transactions on Computer, Vol. 9, No. 3, pp. 353–372, 1997.

15. W. Kim, “A Model of queries for object-oriented databases,” Proceedings International Conference on Very Large Data Bases, 1989.
16. J.B. Lim, A.R. Hurson, L.L. Miller, and Y.C. Chehadeh, “A dynamic clustering scheme for distributed object-oriented databases,” Mathematical Modeling and Scientific Computing, Vol. 8, 1997.
17. NASDAQ World Wide Web Home Page, http:\\www.nadsaq.com, Dec. 1997.
18. A.P. Sheth and J.A. Larson, “Federated database systems for managing distributed, heterogeneous databases,” ACM Computing Surveys, Vol. 22, No. 3, pp. 183–236, 1990.
19. M. Weiser, “Some computer science issues in ubiquitous computing,” Communications of the ACM, Vol. 36, No. 7, pp. 75–84, 1993.
20. S. Zdonik, R. Alonso, M. Franklin, and S. Acharya, “Are disks in the air just pie in the sky?” Proceedings Workshop on Mobile Computing Systems and Applications, 1994, pp. 1–8.

Y.C. Chehadeh received a BS in electrical engineering from University of Arkansas in 1989. He then joined the technical staff at Greenleaf Software in Dallas, TX, for one year, where he was mainly involved in software development for serial communication systems. He began his graduate work at Penn State University in 1990. He received an MS in computer engineering in 1993. In the summer of 1993 he worked on a control-system project for the Kuwait Institute for Scientific Research. From 1985 to 1997, he worked as a development engineer for Honeywell Inc. in Fort Washington, PA, on the development of an embedded distributed control system product. He then returned back full time to Penn State, resumed his graduate work, and received a Ph.D. in computer science and engineering. Dr. Chehadeh is currently working as a member of technical staff for Lucent Technologies’ Optical Networking Group in NJ. He has several refereed technical publications in various conference proceedings and journals. His main research interests are within the area of tele/data communication systems, distributed systems, and databases.

A.R. Hurson is a Computer Science and Engineering Faculty at The Pennsylvania State University. His research for the past 16 years has been directed toward the design and analysis of general as well as special purpose computer architectures. His research has been supported by NSF, NCR Corp., DARPA, IBM, Lockheed Martin, and Penn State University. He has published over 170 technical papers in areas including computer architecture, parallel and distributed processing, dataflow architectures, cache memory, database systems, multidatabases, object oriented databases, and VLSI algorithms. Dr. Hurson served as the Guest Co-Editor of special issues of the *IEEE Proceedings on Supercomputing Technology*, the *Journal of Parallel and Distributed Computing on Load Balancing and Scheduling*, and the *Journal of Integrated Computer-Aided Engineering on Multidatabase and Interoperable Systems*. He is the co-author of the *IEEE Tutorials on Parallel Architectures for Database Systems, Multidatabase Systems: An advanced solution for global information sharing, Parallel Architectures for*

Data/Knowledge Base Systems, and Scheduling and Load Balancing in Parallel and Distributed Systems. He is also the Co-founder of the *IEEE Symposium on Parallel and Distributed Processing (recently merged with IPPS).*

Professor Hurson has been active in various IEEE/ACM Conferences and has given tutorials for various conferences on global information sharing, dataflow processing, database management systems, supercomputer technology, data/knowledge-based systems, scheduling and load balancing, and parallel computing. He served as a member of the IEEE Computer Society Press Editorial Board and an IEEE Distinguished speaker. Currently, he is serving in the IEEE/ACM Computer Sciences Accreditation Board, as the editor of IEEE transactions on computers, and as an ACM lecturer.

Mohsen Kavehrad is a Professor of Electrical Engineering at The Pennsylvania state University. He received his B.Sc. degree in Electronics from Tehran Polytechnic Institute, Iran, in 1973, the M.Sc. degree from Worcester Polytechnic Institute (WPI) in Massachusetts, USA, in 1975 and his Ph.D. degree from Polytechnic University (Formerly: Brooklyn Polytechnic Institute), Brooklyn, New York, in November 1977 in Electrical Engineering. Between 1978 and 1981, he worked for Fairchild Industries (Space Communications Group), GTE Satellite Corp. and GTE Laboratories in Massachusetts. In December 1981 he Joined AT&T Bell Laboratories where he worked in Research, Development, and Systems Engineering areas as a Member of Technical Staff. In March 1989, he joined the Department of Electrical Engineering at University of Ottawa, as a Full Professor where he was at the same time the Director of Broadband Communications Research Laboratory. He was also the leader of Photonic Networks and Systems Thrust and a project leader in the Telecommunications Research Institute of Ontario (TRIO), a project leader in the Canadian Institute for Telecommunications Research (CITR) and the Director of Ottawa-Carleton communications Center for Research (OCCCR). In the summer of 1991, he was a visiting research professor at NTT Laboratories in Japan. In 1996 he spent a six month sabbatical term as a visiting researcher at Northern Telecom (NORTEL), Ottawa.

On January 1, 1997, his 46th birthday, he joined the Department of Electrical Engineering at PENN STATE as W.L. Weiss (AMERITECH-SPONSORED) endowed Chair Professor of Electrical Engineering and Director of the Center for Information and Communications Technology Research (CICTR) at Penn State. He is a consultant to industry. He is also on the Advisory Committee of the Department of Electrical Engineering at Worcester Polytechnic Institutre (WPI) in Worcester, MASS.

He is a former Technical Editor for the IEEE Transactions on Communications, IEEE Communications Magazine and the IEEE Magazine of Lightwave Telecommunications Systems. Presently, he is on the Editorial Board of the International Journal of Wireless Information Networks. He has chaired, organized and been on the advisory committe for several international conferences and workshops. He has worked in the fields of: satellite communications, point-to-point microwave radio communications, portable and mobile radio communications, and atmospheric laser communications. His current research interests are optical fiber communications and networking and broadband local wireless communications. He has published close to 200 papers and holds several issued patents in these areas. He was elected a Fellow of the IEEE in January 1992 for his contributions to Digital Wireless Communications and Optical Systems and Networks. Also, in 1992, he was elected as an IEEE Communications Society "distinguished Speaker." He received 3 Exceptional Technical Contributions awards while working at Bell Laboratories., the 1991 TRIO Feedback award for his patent on a "Passive Optical Interconnect" and 5 best paper awards plus a Canada NSERC Ph.D. Thesis Prize, jointly with his graduate students at University of Ottawa. Furthermore, his name has been listed in several "Who's Who" citations in the World.