Chapter IV

Data Broadcasting in a Mobile Environment

A.R. Hurson, The Pennsylvania State University, USA

Y. Jiao, The Pennsylvania State University, USA

Abstract

The advances in mobile devices and wireless communication techniques have enabled anywhere, anytime data access. Data being accessed can be categorized into three classes: private data, shared data, and public data. Private and shared data are usually accessed through on-demand-based approaches, while public data can be most effectively disseminated using broadcasting. In the mobile computing environment, the characteristics of mobile devices and limitations of wireless communication technology pose challenges on broadcasting strategy as well as data-retrieval method designs. Major research issues include indexing scheme, broadcasting over single and parallel channels, data distribution and replication strategy, conflict resolution, and data retrieval method. In this chapter, we investigate solutions proposed for these issues. High performance and low power consumption are the two main objectives of the proposed schemes. Comprehensive simulation results are used to demonstrate the effectiveness of each solution and compare different approaches.

Introduction

The increasing development and spread of wireless networks and the need for information sharing has created a considerable demand for cooperation among existing, distributed, heterogeneous, and autonomous information sources. The growing diversity in the range of information that is accessible to a user and rapidly expanding technology have changed the traditional notion of timely and reliable access to global information in a distributed system. Remote access to data refers to both mobile nodes and fixed nodes accessing data within a platform characterized by the following:

- low bandwidth,
- frequent disconnection,
- high error rates,
- limited processing resources, and
- limited power sources.

Regardless of the hardware device, connection medium, and type of data accessed, users require timely and reliable access to various types of data that are classified as follows:

- Private data, that is, personal daily schedules, phone numbers, and so forth. The reader of this type of data is the sole owner or user of the data.
- Public data, that is, news, weather information, traffic information, flight information, and so forth. This type of data is maintained by one source and shared by many—a user mainly queries the information source(s).
- Shared data, that is, traditional, replicated, or fragmented databases. Users usually send transactions as well as queries to the information source(s).

  Access requests to these data sources can be on-demand-based or broadcast-based.

On-Demand-Based Requests

In this case users normally obtain information through a dialogue (two-way communication) with the database server—the request is pushed to the system, data sources are accessed, operations are performed, partial results are collected and integrated, and the final result is communicated back to the user. This access scenario requires a solution that addresses the following issues.

- **Security and access control.** Methods that guarantee authorized access to the resources.
- **Isolation.** Means that support operations off-line if an intentional or unintentional disconnection has occurred.

- **Semantic heterogeneity.** Methods that can handle differences in data representation, format, structure, and meaning among information sources and hence establish interoperability.
- **Local autonomy.** Methods that allow different information sources to join and depart the global information-sharing environment at will.
- **Query processing and query optimization.** Methods that can efficiently partition global queries into subqueries and perform optimization techniques.
- **Transaction processing and concurrency control.** Methods that allow simultaneous execution of independent transactions and interleave interrelated transactions in the face of both global and local conflicts.
- **Data integration.** Methods that fuse partial results to draw a global result.
- **Browsing.** Methods that allow the user to search and view the available information without any information processing overhead.
- **Distribution transparency.** Methods to hide the network topology and the placement of the data while maximizing the performance for the overall system.
- **Location transparency.** Methods that allow heterogeneous remote access (HRA) to data sources. Higher degrees of mobility argue for higher degrees of heterogeneous data access.
- **Limited resources.** Methods that accommodate computing devices with limited capabilities.

The literature is abounded with solutions to these issues (Badrinath, 1996; Bright, Hurson, & Pakzad, 1992, 1994; Joseph, Tauber, & Kaashoek, 1997; Satyanarayanan, 1996). Moreover, there are existing mobile applications that address the limited bandwidth issues involved in mobility (Demers, Pertersen, Spreitzer, Terry, Theier, & Welch, 1994; Fox, Gribble, Brewer, & Amir, 1996; Honeyman, Huston, Rees, & Bachmann, 1992; Joseph et al., 1997; Kaashoek, Pinckney, & Tauber, 1995; Lai, Zaslavsky, Martin, & Yeo, 1995; Le, Burghardt, Seshan, & Rabaey, 1995; Satyanarayanan, 1994, 1996).

Broadcast-Based Requests

Public information applications can be characterized by (a) massive numbers of users and (b) the similarity and simplicity in the requests solicited by the users. The reduced bandwidth attributed to the wireless environment places limitations on the rate of the requests. Broadcasting (one-way communication) has been suggested as a possible solution to this limitation. In broadcasting, information is provided to all users of the air channels. Mobile users are capable of searching the air channels and pulling the desired data. The main advantage of broadcasting is that it scales up as the number of users increases and, thus, eliminates the need to multiplex the bandwidth among users accessing the air channel. Furthermore, broadcasting can be considered as an additional storage available over the air for mobile clients. Within the scope of broadcasting one needs to address three issues:

- effective data organization on the broadcast channel,
- efficient data retrieval from the broadcast channel, and
- data selection.

The goal is to achieve high performance (response time) while minimizing energy consumption. Note that the response time is a major source of power consumption at the mobile unit (Imielinski & Badrinath, 1994; Imielinski & Korth, 1996; Imielinski, Viswanathan, & Badrinath, 1997; Weiser, 1993). As a result, the reduction in response time translates into reducing the amount of time a mobile unit spends accessing the channel(s) and thus has its main influence on conserving energy at the mobile unit.

Chapter Organization

In this chapter, we first introduce the necessary background material. Technological limitations are outlined and their effects on the global information-sharing environment are discussed. Issues such as tree-based indexing, signature-based indexing, data replication, broadcasting over single and parallel channels, data distribution, conflict, and data access are enumerated and analyzed next. Then we present solutions to these issues with respect to the network latency, access latency, and power management. Finally, we conclude the chapter and point out some future research directions.

Mobile Computing

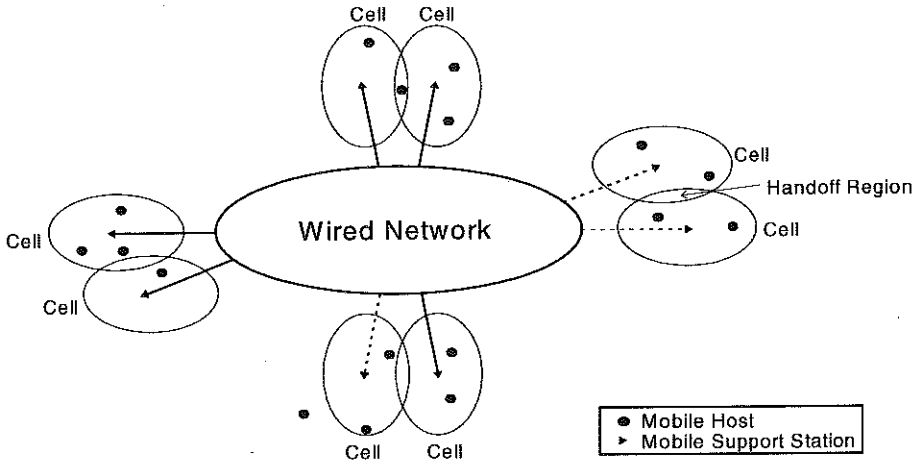
The mobile computing environment is composed of a number of network servers enhanced with wireless transceivers—mobile support stations (MSSs) and a varying number of mobile hosts (MHs) free to move at will (Figure 1).

The role of the MSS is to provide a link between the wireless network and the wired network. The link between an MSS and the wired network could be either wireless (shown as a dashed line) or wire based. The area covered by the individual transceiver is referred to as a cell. To satisfy a request, an MH accesses the MSS responsible for the cell where the MH is currently located. It is the duty of the MSS to resolve the request and deliver the result back to the client. Once an MH moves across the boundaries of two cells, a handoff process takes place between the MSSs of the corresponding cells. The MH is normally small, lightweight, and portable. It is designed to be compact with limited resources relying on temporary power supplies (such as batteries) as its main power source.

Characteristics of the Mobile Environment

Wireless communication is accomplished via modulating radio waves or pulsing infrared light. Table 1 summarizes a variety of mobile network architectures. Mainly, three

Figure 1. Architecture of the mobile-computing environment



characteristics distinguish the mobile computing environment from traditional wired computing platforms, namely, wireless medium, mobility, and portability.

Wireless Medium

The common ground among all wireless systems is the fact that communication is done via the air (and not via cables). This fact changes a major underlying assumption behind the conventional distributed algorithms. The physical layer of the connection is no longer the reliable coaxial or optic cable. Communication over the air is identified by frequent disconnections, low data-rate, high cost, and lack of security (Alonso & Ganguly, 1992; Alonso & Korth, 1993; Chlamtac & Lin, 1997; Imielinski & Badrinath, 1994; Imielinski & Korth, 1996; Imielinski et al., 1997; Weiser, 1993).

Mobility

Mobility introduces new challenges beyond the scope of the traditional environment. Mobile devices can be used at multiple locations and in transition between these locations. Mobility results in several issues including disconnections due to handoff processes, motion management, location-dependent information, heterogeneous and fragmented networks, security, and privacy.

Portability

There are many variations of portable computer systems with different physical capabilities. However, they share many common characteristics such as limited memory,

processing power, and power source. The ideal goal would be to develop a device that is compact, durable, lightweight, and that consumes a minimum amount of power. Table 2 highlights some limitations of the mobile environment.

Broadcasting

The cost of communication is normally asymmetric: Sending information requires 2 to 10 times more energy than receiving the information (Imielinski, Viswanathan, & Badrinath, 1994). In the case of accessing public information, instead of the two-way, on-demand, traditional communication pattern, popular public information can be generated and disseminated over the air channel. The MH requiring the information can tune to the broadcast and access the desired information from the air channel.

In general, data can be broadcast either on one or several channels. Broadcasting has been used extensively in multiple disciplines, that is, management of communication systems (Comer, 1991) and distributed database environments (Bowen, 1992). In this chapter, the term *broadcast* is referred to as the set of all broadcast data elements (the stream of data across all channels). 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.

In the data-broadcasting application domain, power consumption and network latency are proven constraints that limit "timely and reliable" access to information. The necessity of minimizing power consumption and network latency lies in the limitation of current technology. The hardware of the mobile units have been designed to mitigate this

Table 1. Mobile network architectures

Architecture	Description
Cellular Networks	• Provides voice and data services to users with handheld phones • Continuous coverage is restricted to metropolitan regions • Movement over a wide area may need user to inform the network of the new location • Low bandwidth for data-intensive applications • Could be based on either analog technology or digital technology
Wireless LANs	• A traditional LAN extended with a wireless interface • Serves small, low-powered, portable terminals capable of wireless access • Connected to a more extensive backbone network, such as a LAN or WAN
Wide Area Wireless Networks	• Special mobile radio networks provided by private service providers (RAM, ARDIS) • Provides nationwide wireless coverage for low-bandwidth data services, including e-mail or access to applications running on a fixed host
Paging Networks	• Receive-only network • No coverage problems • Low bandwidth • Unreliable

Satellite Networks	• Unlike the static, grounded MSSs, satellites are not fixed
	• Normally classified based on their altitudes (from earth) into three classes:
	  • Low Earth Orbit Satellites (LEOS)
	  • Medium Earth Orbit Satellites (MEOS)
	  • Geostationary Satellites (GEOS)

Table 2. Limitations of the mobile environment

Limitations	Concerns/Side Effects
Frequent Disconnections	• Handoff blank out in cellular networks • Long down time of the mobile unit due to limited battery power • Voluntary disconnection by the user • Disconnection due to hostile events (e.g., theft, destruction) • Roaming off outside the geographical coverage area of the window service
Limited Communication Bandwidth	• Quality of service (QoS) and performance guarantees • Throughput and response time and their variances • Efficient battery use during long communication delays
Heterogeneous and Fragmented Wireless Network Infrastructure	• Rapid and large fluctuations in network QoS • Mobility transparent applications perform poorly without mobility middleware or proxy • Poor end-to-end performance of different transport protocols across network of different parameters and transmission characteristics

limitation by operating in various operational modes such as active, doze, sleep, nap, and so forth to conserve energy. A mobile unit can be in active mode (maximum power consumption) while it is searching or accessing data; otherwise, it can be in doze mode (reduced power consumption) when the unit is not performing any computation. Along with the architectural and hardware enhancements, efficient power management and energy-aware algorithms can be devised to manage power resources more effectively. In addition, appropriate retrieval protocols can be developed to remedy network latency and hence to allow faster access to the information sources. In general, two issues need to be considered.

- The MH should not waste its energy in continuously monitoring the broadcast to search for information. As a result, the information on the broadcast should be organized based on a disciplined order. Techniques should be developed to (a) instruct the MH of the availability of the data element on the broadcast and (b) if the data element is available, instruct the MH of the location of the data element on the broadcast.
- An attempt should be made to minimize the response time. As will be seen later, this is achieved by shortening the broadcast length and/or reducing the number of passes over the air channel(s).

Data Organization on the Air Channel

Unlike the conventional wired environment, where a disk is assumed to be the underlying storage, data in the mobile environment are stored on air channel(s). A disk and an air channel have major structural and functional differences. The disk has a three-dimen-

sional structure (disks can have a four-dimensional structure if multiple disks are used — redundant arrays of independent disks [RAID]). An air channel, on the other hand, is a one-dimensional structure. The disk has a random-access feature and the air channel is sequential in nature. Finally, the current raw data rate of a disk is generally much higher than that of the air channel.

Zdonik, Alonso, Franklin, and Acharya (1994) and Acharya, Alonso, Franklin, and Zdonik (1995) investigated the mapping of disk pages onto a broadcast channel and the effects of that mapping on the management of cache at the MH. In order to place disk pages onto the data channel, the notion of multiple disks with different sizes spinning at multiple speeds was used. Pages available on faster spinning disks get mapped more frequently than those available on slower disks. In cache management, a nonconventional replacement strategy was suggested. Such a policy assumed that the page to be replaced might not be the least-recently used page in the cache. This is justifiable since the set of pages that are most frequently in demand are also the most frequently broadcast. This work was also extended to study the effect of prefetching from the air channel into the cache of the MH. These efforts assumed the same granularity for the data items on air channel and disk pages: if a data item is to be broadcast more frequently (replicated), the entire page has to be replicated. In addition, due to the plain structural nature of the page-based environment, the research looked at the pages as abstract entities and was not meant to consider the contents of the pages (data and its semantics) as a means to order the pages. In object-oriented systems, semantics among objects greatly influence the method in which objects are retrieved and, thus, have their direct impact on the ordering of these objects or pages. In addition, the replication should be performed at the data item granularity level.

An index is a mechanism that speeds up associative searching. An index can be formally defined as a function that takes a key value and provides an address referring to the location of the associated data. Its main advantage lies in the fact that it eliminates the need for an exhaustive search through the pages of data on the storage medium. Similarly, within the scope of broadcasting, an index points to the location or possible availability of a data item on the broadcast, hence, allowing the mobile unit to predict the arrival time of the data item requested. The prediction of the arrival time enables the mobile unit to switch its operational mode into an energy-saving mode. As a result, an indexing mechanism facilitates data retrieval from the air channel(s), minimizing response time

Table 3. Advantages and disadvantages of indexing schemes

Advantages	Disadvantages
Provides auxiliary information that allows mobile users to predict arrival time of objects	Longer broadcast
Enables utilization of different operational modes (active, nap, doze, etc.)	Longer response time
Reduces power consumption (less tune-in time)	Computational overhead due to complexity in retrieval, allocation, and maintenance of the indexes

while reducing power consumption. Table 3 summarizes the advantages and disadvantages of indexing schemes.

The literature has addressed several indexing techniques for a single broadcast channel as well as parallel broadcast channels with special attention to signature-based indexing and tree-based indexing (Boonsiriwattanakul, Hurson, Vijaykrishnan, & Chehadeh, 1999; Chehadeh, Hurson, & Miller, 2000; Chehadeh, Hurson, & Tavangarian, 2001; Hu & Lee, 2000, 2001; Imielinski et al., 1997; Juran, Hurson, & Vijaykrishnan, 2004; Lee, 1996).

Signature-Based Indexing

A signature is an abstraction of the information stored in a record or a file. The basic idea behind signatures on a broadcast channel is to add a control part to the contents of an information frame (Hu & Lee, 2000, 2001; Lee, 1996). This is done by applying a hash function to the contents of the information frame, generating a bit vector, and then superimposing it on the data frame. As a result, a signature partially reflects the data content of a frame. Different allocations of signatures on a broadcast channel have been studied; among them, three policies, namely, *single signature*, *integrated signature*, and *multilevel signature*, are studied in Hu and Lee (2000) and Lee (1996).

During the retrieval, a query is resolved by generating a signature based on the user's request. The query signature is then compared against the signatures of the data frames in the broadcast. A successful match indicates a possible hit. Consequently, the content of the corresponding information frame is checked against the query to verify that it corresponds to the user's demands. If the data of the frame corresponds to the user's request, the data is recovered; otherwise, the corresponding information frame is ignored. In general, this scheme reduces the access time and the tune-in time when pulling information from the air channel.

Tree-Based Indexing

Two kinds of frames are broadcast on the air channel: data frames and index frames. The index frame contains auxiliary information representing one or several data attributes pointing to the location of data collection (i.e., information frames) sharing the same common attribute value(s). This information is usually organized as a tree in which the lowest level of the tree points to the location of the information frames on the broadcast channel.

A broadcast channel is a sequential medium and, hence, to reduce the mobile unit's active and tune-in time, and consequently to reduce the power consumption, the index frames are usually replicated and interleaved with the data frames. Two index replication schemes (namely, *distributed indexing* and *(1, m) indexing*) have been studied in Imielinski et al. (1997). In distributed indexing, the index is partitioned and interleaved in the broadcast cycle (Hu & Lee, 2000, 2001; Lee, 1996). Each part of the index in the broadcast is followed by its corresponding data frame(s). In (1, m) indexing, the entire index is interleaved m times during the broadcast cycle (Imielinski et al., 1997; Lee 1996) — the whole index is broadcast before every $1/m$ fraction of the cycle.

Previous work has shown that the tree-based indexing schemes are more suitable for applications where information is accessed from the broadcast channel randomly, and the signature-based indexing schemes are more suitable in retrieving sequentially structured data elements (Hu & Lee, 2000, 2001). In addition, tree-based indexing schemes have shown superiority over the signature-based indexing schemes when the user request is directed towards interrelated objects clustered on the broadcast channel(s). Furthermore, tree-based indexing schemes relative to signature-based indexing schemes are more suitable in reducing the overall power consumption. This is due to the fact that a tree-based indexing provides global information regarding the physical location of the data frames on the broadcast channel. On the other hand, signature-based indexing schemes are more effective in retrieving data frames based on multiple attributes (Hu & Lee, 2000). Table 4 compares and contrasts the signature- and tree-based indexing.

Data Organization on a Single Channel

An appropriate data placement algorithm should attempt to detect data locality and cluster related data close to one another. An object-clustering algorithm takes advantage of semantic links among objects and attempts to map a complex object into a linear sequence of objects along these semantic links. It has been shown that such clustering can improve the response time by an order of magnitude (Banerjee, Kim, Kim, & Garza, 1988; Chang & Katz, 1989; Chehadeh, Hurson, Miller, Pakzad, & Jamoussi, 1993; Cheng & Hurson, 1991a). In the conventional computing environment, where data items are stored on disk(s), the clustering algorithms are intended to place semantically connected objects physically along the sectors of the disk(s) close to one another (Cheng & Hurson, 1991a). The employment of broadcasting in the mobile computing environment motivates the need to study the proper data organization along the sequential air channel. Figure 2 depicts a weighted directed acyclic graph (DAG) and the resulting clustering sequences achieved when different clustering techniques are applied.

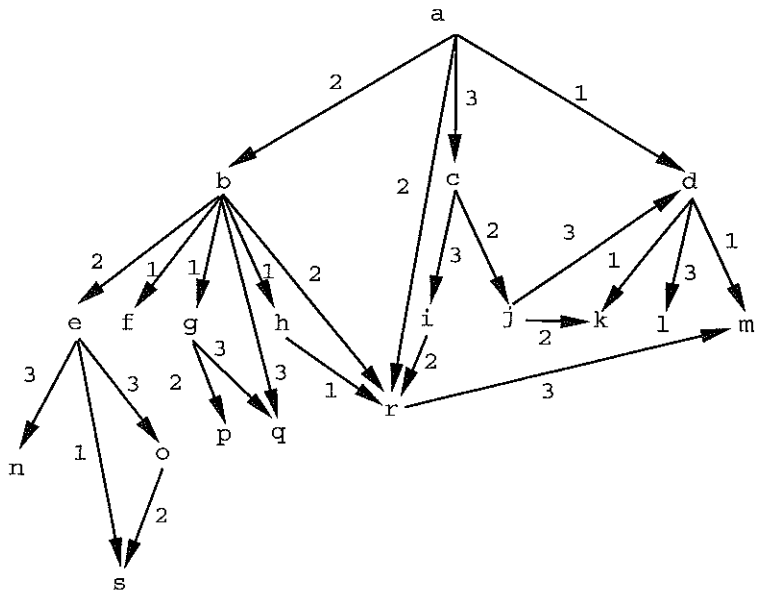
Table 4. Signature-based versus tree-based indexing

Feature	Signature-Based Indexing	Tree-Based Indexing
Less power consumption		✓
Longer length of broadcast	✓	✓
Computational overhead	✓	✓
Longer response time	✓	✓
Shorter tune-in time		✓
Random data access		✓
Sequentially structured data	✓	
Clustered data retrieval		✓
Multi-attribute retrieval	✓	

In order to reduce the response time, the organization of data items on an air channel has to meet the following three criteria.

- **Linear ordering.** The one-dimensional sequential access structure of the air channel requires that the object ordering be linear. In a DAG representation of a complex object, an edge between two nodes could signify an access pattern among the two nodes. The *linearity* property is defined as follows: 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.
- **Minimum linear distance between related objects.** In a query, multiple objects might be retrieved following their connection patterns. Intuitively, reducing the distance among these objects along the broadcast reduces the response time and power consumption.
- **More availability for popular objects.** In a database, not all objects are accessed with the same frequency. Generally, requests for 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.

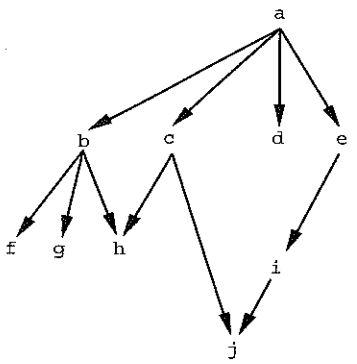
Figure 2. Graph and various clustering methods



Clustering Method	Resulting Sequence
Depth First	abensofgpqhrmcijdkl
Breadth First	abcdefgqhijklmnsop
Children-Depth First	abrcdefgqhmnsopijkl
Level Clustering	acibgqprmenosjdlkfh

Figure 3 depicts a directed graph and multiple linear sequences that satisfy the linear ordering property. The 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, a data unit is used as a unit of measurement. Furthermore, it is assumed that all data items are of equal size. The cost associated with an edge between a pair of data items is calculated by counting the number of data items that separate these two in the linear sequence. For example, in the abfgchdeij sequence, data items a and d are separated by the sequence bfgch and thus have a cost of 6. The rightmost 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 reduced average linear distance translates into smaller average response time. In this example, the best linear sequence achieves a total sum of 26.

Figure 3. Graph, linear sequences, and costs



	Linear Sequence	Individual Costs											Total Cost
		ab	ac	ad	ae	bf	bg	bh	ch	cj	ei	ij	
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

Data Organization on Parallel Channels

The broadcast length is a factor that affects the average response time in retrieving data items from the air channel — reducing the broadcast length could also reduce the response time. The broadcast length can be reduced if data items are broadcast along parallel air channels.

Formally, we attempt to assign the objects from a weighted DAG onto multiple channels, while (a) preserving dependency implied by the edges, (b) minimizing the overall broadcast time (load balancing), and (c) clustering related objects close to one another (improving the response time). As one could conclude, there are trade-offs between the second and third requirements: Achieving load balancing does not necessarily reduce the response time in accessing a series of data items.

Assuming that all channels have the same data rate, one can draw many analogies between this problem and static task scheduling in a homogeneous multiprocessor environment — tasks are represented as a directed graph $D \equiv (N, A)$, with nodes (N) and directed edges (A) representing processes and dependence among the processes, respectively. Compared to our environment, channels can be perceived as processors (PEs), objects as tasks, and the size of a data item as the processing cost of a task. There is, however, a major distinction between the two environments. In the multiprocessor environment, information is normally communicated among the PEs, while in the multi-channel environment there is no data communication among channels.

The minimum makespan problem, in static scheduling within a multiprocessor environment, attempts to find the minimum time in which n dependent tasks can be completed on m PEs. An optimal solution to such a problem is proven to be NP hard. Techniques such as graph reduction, max-flow min-cut, domain decomposition, and priority list scheduling have been used in search of suboptimal solutions. Similar techniques can be developed to assign interrelated objects closely over parallel channels.

Distribution of data items over the broadcast parallel air channels brings the issue of access conflicts between requested data items that are distributed among different channels. The access conflict is due to two factors:

- the receiver at the mobile host can only tune into one channel at any given time, and
- the time delay to switch from one channel to another.

Access conflicts require the receiver to wait until the next broadcast cycle(s) to retrieve the requested information. Naturally, multiple passes over the broadcast channels will have a significant adverse impact on the response time and power consumption.

Conflicts in Parallel Air Channels

Definition 1. A K-data item request is an application request intended to retrieve K data items from a broadcast.

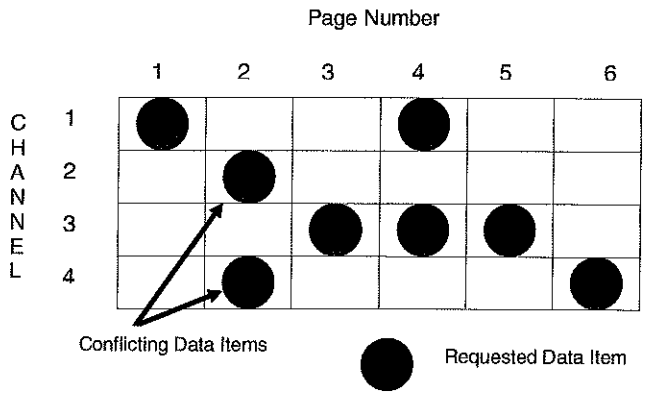
It is assumed that each channel has the same number of pages (frames) of equal length and, without loss of generality, each data item is residing on only a single page. A single broadcast can be modeled as an N x M grid, where N is the number of pages per broadcast and M is the number of channels. In this grid, K data items ($0 \leq K \leq MN$) are randomly distributed throughout the MN positions of the grid. Based on the common page size and the network speed, the time required to switch from one channel to another is equivalent to the time it takes for one page to pass in the broadcast. Thus, it is impossible for the mobile unit to retrieve both the ith page on Channel A and $(i + 1)$th page on Channel B (where A $\neq$ B). Figure 4 is a grid model that illustrates this issue.

Definition 2. Two data items are defined to be in conflict if it is impossible to retrieve both on the same broadcast.

In response to a user request, the access latency is then directly dependent on the number of passes over the broadcast channels. One method of calculating the number of required passes over the broadcast channels is to analyze the conflicts between data items. For any particular data item, all data items in the same or succeeding page (column) and on a different row (channel) will be in conflict. Thus, for any specific page (data object) in the grid, there are $(2M - 2)$ conflicting pages (data items) in the broadcast (The last column has only $M - 1$ conflict positions, but it is assumed that N is sufficiently large to make this difference insignificant.) These $(2M - 2)$ positions are known as the conflict region.

For any particular data item, it is possible to determine the probability of exactly i conflicts occurring, or $P(i)$. Because the number of conflicts for any particular data item is bounded by $(M - 1)$, the weighted average of these probabilities can be determined by summing a finite series. This weighted average is the number of broadcasts (passes) required to retrieve all K data items if all conflicts between data items are independent.

Figure 4. Sample broadcast with $M = 4$, $N = 6$, and $K = 8$



$$B = \sum_{i=0}^{M-1} (i+1) * P(i) \quad (1)$$

Access Patterns

In order to reduce the impact of conflicts on the access time and power consumption, retrieval procedures should be enhanced by a scheduling protocol that determines data retrieval sequence during each broadcast cycle. The scheduling protocol we proposed is based on the following three prioritized heuristics:

1) Eliminate the number of conflicts
2) Retrieve the maximum number of data items
3) Minimize the number of channel switches

The scheme determines the order of retrieval utilizing a forest - an *access forest*. An access forest is a collection of trees (*access trees*), where each access tree represents a collection of access patterns during a broadcast cycle. Naturally, the structure of the access forest, that is, the number of trees and the number of children that any parent can have, is a function of the number of broadcast channels.

Definition 3. An access tree is composed of two elements: nodes and arcs.

- **Node.** A node represents a requested data item. The nodes are labeled to indicate its conflict status: mnemonically, C_1 represents when the data item is in conflict with another data item(s) in the broadcast and C_0 indicates the lack of conflict.

Each access tree in the access forest has a different node as a root-the root is the first accessible requested data item on a broadcast cycle. This simply implies that an access forest can have at most n trees where n is the number of broadcast channels.

- **Arcs.** The arcs of the trees are weighted arcs. A weight denotes whether or not channel switching is required in order to retrieve the next scheduled data item in the access pattern. A branch in a tree represents a possible access pattern of data items during a broadcast cycle with no conflicts. Starting from the root, the total number of branches in the tree represents all possible access patterns during a broadcast cycle.

This scheme allows one to generate all possible nonconflicting, weighted access patterns from all channels. The generated access patterns are ranked based on their weights-a weight is set based on the number of channel switches-and then the one(s) that allows the maximum number of data retrievals with minimum number of channel switches is selected. It should be noted that the time needed to build and traverse the access forest is a critical factor that must be taken into account to justify the validity of

this approach. The following working example provides a detailed guide to illustrate the generation of the access patterns for each broadcast cycle.

1) **Search.** Based on the user's query, this step determines the offset and the channel number(s) of the requested objects on the broadcast channels. Figure 5 depicts a request for eight data items from a parallel broadcast channel of four channels.

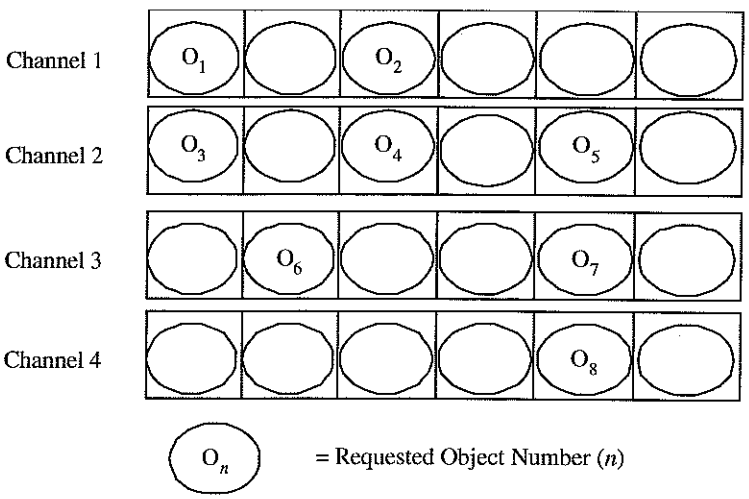
2) **Generation of the access forest.** For each broadcast channel, search for the requested data item with the smallest offset (these objects represent the roots of an access tree). For the example, the data items with the smallest offsets are O_1, O_3, O_6 and O_8. Note that the number of access trees is upper bounded by the number of broadcast channels.

3) **Root assignment.** For each channel with at least one data item requested, generate a tree with root node as determined in Step 2. The roots are temporarily tagged as C_0.

4) **Child assignment.** Once the roots are determined, it is necessary to select the child or children of each rooted access tree: For each root, and relative to its position on the air channel, the algorithm determines the closest nonconflicting data items on each channel. With respect to a data item $O_{i,x}$ at location X on air channel i ($1 \le i \le n$), the closest nonconflicting data item is either the data item $O_{i,x+1}$ or the data item $O_{j,x+2}$, $j \neq i$. If the child is in the same broadcast channel as the root, the arc is weighted as 0; otherwise it is weighted as 1. Each added node is temporarily tagged as C_0. Figure 6 shows a snapshot of the example after this step.

5) **Root label update.** Once the whole set of requested data items is analyzed and the access forest is generated, the conflict labels of the nodes of each tree are updated. This process starts with the root of each tree. If a root is in conflict with any other

Figure 5. A parallel broadcast of four channels with eight requested data items



root(s), a label of C_1 is assigned to all the roots involved in the conflict, otherwise the preset value of C_0 is maintained.

6) **Node label update.** Step 5 will be applied to the nodes in the same level of each access tree in the access forest. As in Step 5, a value of C_1 is assigned to the nodes in conflict. Figure 7 shows the example with the updated labels.

7) **Sequence selection.** The generation of the access forest then allows the selection of the suitable access patterns in an attempt to reduce the network latency and power consumption. A suitable access pattern is equivalent to the selection of a tree branch that:
   - has the most conflicts with other branches,
   - allows more data items to be pulled off the air channels, and
   - requires the least number of channel switches.

The O_3, O_4, and O_5 sequence represents a suitable access pattern for our running example during the first broadcast cycle. Step 7 will be repeated to generate access patterns for different broadcast cycles. The algorithm terminates when all the requested data items are covered in different access patterns. The data item sequence O_1, O_2, and O_7 and data

Figure 6. Children of each root

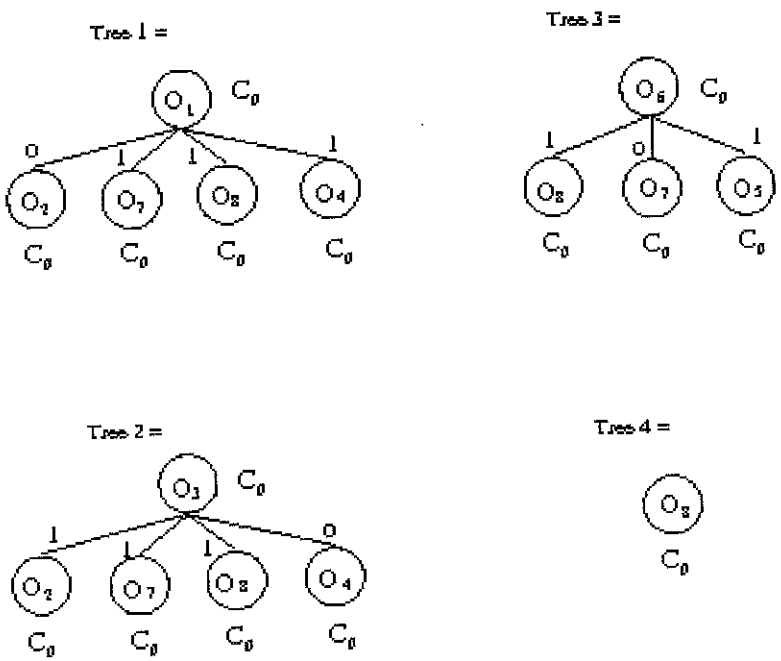
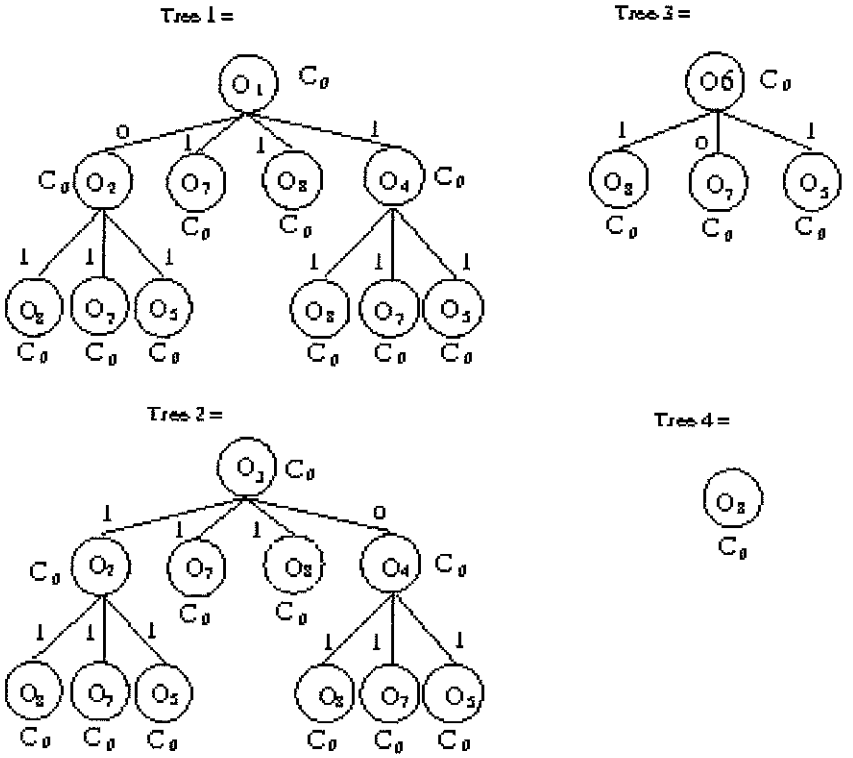


Figure 7. Final state of the access forest



item sequence O_6 and O_8 represent the last two patterns for retrieving all of the data items requested in the example.

Data Organization on a Single Channel

As noted in the literature, the object-oriented paradigm is a suitable methodology for modeling public data that are by their very nature in multimedia format (Atkinson, Bancilhon, DeWitt, Dittrich, Maier, & Zdonik, 1989; Fong, Kent, Moore, & Thompson, 1991; Hurson, Pakzad, & Cheng, 1993; Kim, 1990). In addition, object-oriented methodology provides a systematic mechanism to model a complex object in terms of its simpler components.

In this section, without loss of generality, we model information units as objects. Object clustering has proven to be an effective means of data allocation that can reduce response times (Banerjee et al., 1988; Chang & Katz 1989; Chehadeh et al., 1993; Cheng & Hurson, 1991b; Lim, Hurson, Miller, & Chehadeh, 1997). In our research, we investi-

gated two heuristic allocation strategies. The first strategy assumes a strict linearity requirement and deals with nonweighted DAGs. The second approach relaxes such restriction in favor of clustering strongly related objects closer to one another and consequently deals with weighted DAGs.

Strict Linearity: ApproximateLinearOrder Algorithm

Definition 4. An independent node is a node that has either one or no parent. A graph containing only independent nodes makes up a forest.

Heuristic Rules

1) Order the children of a node based on their number of descendants in ascending order.
2) Once a node is selected, all of its descendants should be visited and placed on the sequence in a depth-first manner, without any interruptions from breadth siblings.
3) If a node has a nonindependent child, with all of its parents already visited, the nonindependent child should be inserted in the linear sequence before any independent child.

The ApproximateLinearOrder algorithm implements these heuristics and summarizes the sequence of operations required to obtain 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.

ApproximateLinearOrder Algorithm

1) traverse DAG using DFS traversal and as each node is traversed
2) append the traversed node N to the sequence
3) remove N from {nodes to be traversed}
4) **if** {nonindependent children of N having all their parents in the sequence} $\neq \varnothing$
5) $Set \leftarrow$ {nonindependent children of N having all their parents in the sequence}
6) **else**
7) **if** {independent children of N} $\neq \varnothing$
8) $Set \leftarrow$ {independent children of N}
9) $NextNode \leftarrow$ node $\in Set \mid$ node has least # of descendants among the nodes in Set

Applying this algorithm to the graph of Figure 3 generates either the 5th or 11th sequence — dependent on whether c or d was chosen first as the child with the least number of

independent 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 zero — are considered harmful and thus are not 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.

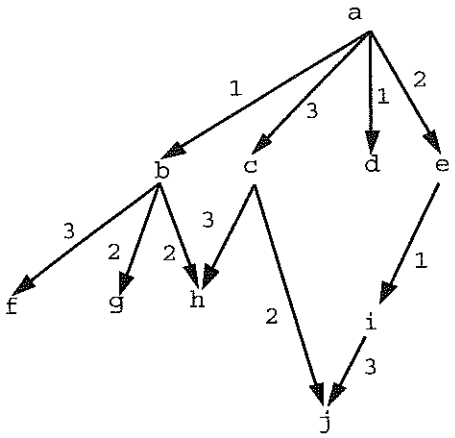
Varying Levels of Connectivity: PartiallyLinearOrder Algorithm

In a complex object, objects are connected through semantic links with different degrees of connectivity. The different access frequency of objects in an object-oriented database reveals that some patterns are more frequently traversed than others (Fong et al., 1991). This observation resulted in the so-called PartiallyLinearOrder algorithm that assumes a weighted DAG as its input and produces a linear sequence. It combines the nodes (single_node) of the graph into multi_nodes in descending order of their connectivity (semantic links). The insertion of single_nodes within a multi_node respects 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. Figure 8 shows the application of the PartiallyLinearOrder algorithm.

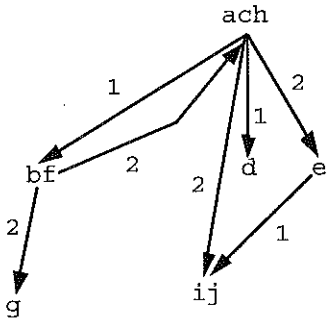
PartiallyLinearOrder Algorithm

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
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 $WeightedLinearDistance_{MN\_AN}$ & $WeightedLinearDistance_{AN\_MN}$
10) merge *MN* & *AN* into one multi_node, based on the appropriate direction

Figure 8. Process of PartiallyLinearOrder



(a) Original Graph



(b) First and Second Iterations

bfgacheijd

(c) Third Iteration

Performance Evaluation

Parameters

A simulator was developed to study the behavior of the proposed mapping algorithms based on a set of rich statistical parameters. Our test bed was an object-oriented financial database. The OO7 benchmark was chosen to generate the access pattern graphs. We used the NASDAQ exchange (NASDAQ, 2002) as our base model, where data is in both textual and multimedia (graphics — i.e., graphs and tables) formats. Table 5 shows 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 6 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.

Results

The simulator operates in two stages.

- Structuring the access-pattern object graph, based on certain statistical parameters, and mapping it along the air channel using various mapping algorithms. To get a wide spectrum of possible graphs, parameters such as (a) the percentage of nonfree nodes, (b) the depths of the trees within the graph, and (c) the amount of sharing that exists between trees through nonfree nodes that were varied. Varying these statistical parameters, we generated 500 access graphs that were used as part of our test bed. In addition, we simulated three mapping algorithms: a nonlinear, children-depth-first clustering algorithm (Banerjee et al., 1988), and the PartiallyLinearOrder and ApproximateLinearOrder algorithms.
- Generating queries and accessing the requested objects from the air channel. During each run, each query on average accesses 20 objects either through their semantic links or randomly (following the [C:R] value of the next-node ratio). The simulator measured the average access delay. Each point in the curves (Figure 9) is the average result of running the simulator 100,000 times. Finally, we assumed a broadcast data rate of 1Mbit/sec and showed the results in terms of seconds.

Impact of Number of Objects. ApproximateLinearOrder and PartiallyLinearOrder schemes performed better than the conventional children-depth first by taking the linearity issue into consideration. As expected 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 retrievals of two consecutive objects. Taking a closer look at this effect, we observed that this extra delay is mainly due to an increase in the distance for objects that are retrieved randomly (not based on their

Table 5. Description of parameters

Parameter	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 (random or connection)
Out-Degree Distribution	Distribution of the type of nodes based on their out-degrees
Level Distribution	Semantic connectivity of 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 6. Input parameter values

Parameter	Default Value	Ranges
Number of Nodes	5,000	400-8,000
Object Size (in Bytes) • Small • Medium • Large	 $2 \leq o < 20$ $20 \leq o < 7K$ $7K \leq o < 50K$	 2-20 20-7K 7K-50K
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

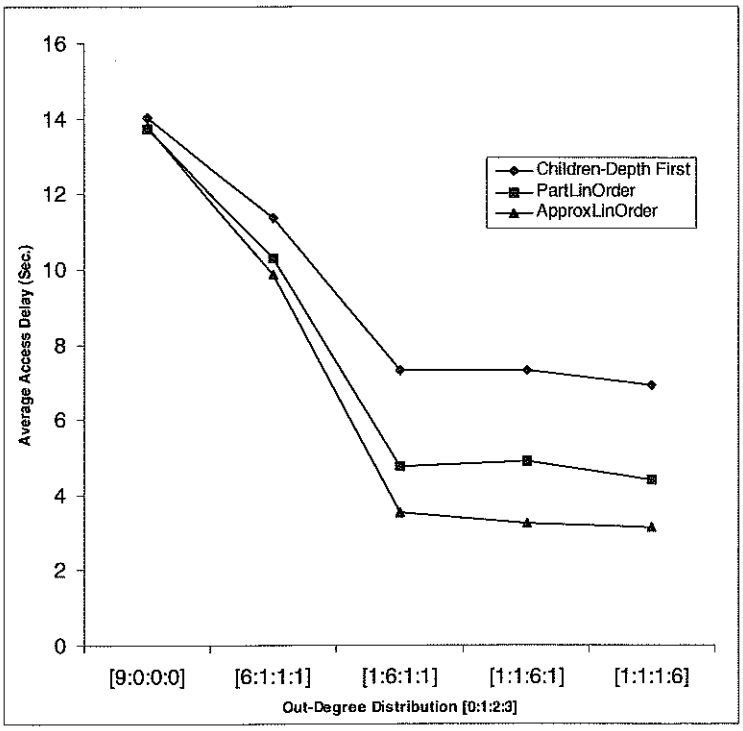
semantic links) since the goal of both algorithms is to cluster semantically related objects close to one another. The ApproximateLinearOrder algorithm outperformed the PartiallyLinearOrder algorithm since the latter attempts to cluster strongly connected objects closer to one another than loosely connected ones and, hence, compromises the linearity property for the loosely connected objects. This compromise overshadows the benefit and is amplified as the number of objects increased. To get a better insight on how our proposed schemes compare with the optimal case, two graphs with 10 nodes were constructed and the optimal sequences exhaustively generated. Using the same set of input values, the average access delay for both proposed schemes were simulated and compared against the average access delay for the optimal sequence. The results of ApproximateLinearOrder and PartiallyLinearOrder were 79% and 76%, respectively, of the access delay of the optimal case.

Size Distribution. In this experiment, we observed that the smallest average access delay took place when the air channel contained smaller data items. However, as the population of data items shifted toward the larger ones, the average access delay increased.

Next-Node Ratio. During the course of a query, objects are either accessed along the semantic links or in a random fashion. At one extreme, when all objects were accessed along the semantic links, the average access delay was minimal. The delay, however, increased for randomly accessed objects. Finally, 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. This parameter indicates the number of children of a node within the graph — an out-degree of 0 indicates a sink node. Figure 9 shows the effect of varying the out-degree distribution within the graph structure. The point [9:0:0:0] indicates that all the nodes within the graph have an out-degree of 0, with 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 it would be more

Figure 9. Average access delay versus connectivity



desirable to deal with more, but simpler, objects than with few complex objects on the air channel.

In separate simulation runs, the simulator was also used to measure the effect of varying the percentage of popular objects and the replication frequency. These two parameters have the same effect on the total number of objects on the air channel, however, from the access pattern perspective, the semantic of the accesses are different. In both cases, the average access delay increased as either parameter increased. We also observed and measured the average access delay for different degrees of connectivity among objects. The average access delay for objects connected through strong connections is about 4.3 seconds, whereas it is 7.3 and 7.6 seconds for normally and weakly connected objects, respectively. As would be expected, these results show that the improvement is considerable for the objects connected by a strong connection, but for a normal connection, the performance was close to that of the weak-connection case since the algorithm performs its best optimization for strongly connected objects.

Section Conclusion

In this section, two heuristically based mapping algorithms were discussed, simulated, and analyzed. Performing the mapping in polynomial time was one of the major issues of concern while satisfying linearity, locality, and replication of popular objects. The ApproximateLinearOrder algorithm is a greedy-based approximation algorithm that guarantees the linearity property and provides a solution in polynomial time. The PartiallyLinearOrder algorithm guarantees the linearity property for the strongest related objects and relaxes the linearity requirement for objects connected through looser links. Finally, it was shown that the proposed algorithms offer higher performance than the traditional children-depth-first algorithm.

Data Organization on Parallel Channels

Reducing the broadcast length is one way to satisfy timely access to the information. This could be achieved by broadcasting data items along parallel air channels. This problem can be stated formally as follows: Assign the data items from a weighted DAG onto multiple channels while (a) preserving dependency implied by the edges, (b) minimizing the overall broadcast time (load balancing), and (c) clustering related data items close to one another (improving the response time). Realizing the similarities between these objectives and the task-scheduling problem in a multiprocessor environment, we proposed two heuristic–based, static scheduling algorithms, namely the largest object first (LOF) algorithm and the clustering critical-path (CCP) algorithm.

The Largest Object First Algorithm

This algorithm relies on a simple and localized heuristic by giving priority to larger data items. The algorithm follows the following procedure: For each collection of data items, recursively, a "proper" node with in-degree of 0 is chosen and assigned to a "proper" channel; a "proper" channel is the one with the smallest overall size and a "proper" node is the largest node with in-degree of 0. The assigned node along with all of its out-edges are eliminated from the object DAG. This results in a set of nodes with in-degree of 0. These nodes are added to the list of free nodes and then are selected based on their sizes. This process is repeated until all the nodes of the DAG are assigned.

Definition 5. A free node is a node that either has an in-degree of 0 (no parent) or has all of its parents allocated on a channel. A free node is a candidate node available for allocation.

Assuming that there are n nodes in the graph, the algorithm requires the traversal of all the nodes and thus requires n steps. At each step, the algorithm searches for the largest

available node whose parents have been fully allocated. This would require at most $O(n^2)$. Therefore, the overall running time of the algorithm is $O(n^3)$. The LOF algorithm respects the dependency among the nodes, if any, and achieves a better load balancing by choosing the largest object first. This algorithm, however, does not allocate objects based on the degree of connectivity and/or the total size of the descendent objects that could play a significant role in balancing the loads on the channels. In addition, this algorithm does not necessarily cluster related object on the parallel air channels.

LOF Algorithm

1) **repeat (2-4)** until all nodes are assigned
2) assign a free node with the largest weight whose parents are fully allocated to the least-loaded channel
3) remove all out-edges of the assigned node from the DAG
4) insert resulting free nodes into the list of free nodes

The Clustering Critical-Path Algorithm

A critical path is defined as the longest sequence of dependent objects that are accessed serially. A critical path is determined based on the weights assigned to each node. A weight is defined based on several parameters such as the size of the data item, the maximum weight of the descendents, the total weight, and the number of descendents.

Definition 6. A critical node is a node that has a child with an in-degree greater than 1.

Load Balancing

Critical Node effect. Allocate a critical node with the highest number of children with in-degrees greater than 1 first.

Number of children with in-degrees of 1. Allocate nodes with the highest number of children with in-degrees of 1 first. This could free up more nodes to be allocated in parallel channels.

Clustering Related Objects

The weight of a node should be made a function of the weights of the incoming and outgoing edges. The weight of each node is calculated based on Equation 2. It should be noted that:

- There is a trade-off between load balancing and clustering related objects: The allocation strategy for the purpose of load balancing could upset the clustering of related objects and vice versa. Therefore, we propose a factor to balance the two

requirements. This factor takes a constant value $\in [0,1]$ and can be assigned to favor either requirement over the other.

- The size of a data item is a multiple of a constant value.
- The weight of an edge is a multiple of a constant value.

$$W = MWC + F\left[S + NCID1 + \sum_{i=1}^{NCIDM} SPC_i - NCIDM(S)\right] + (1-F)\left[(NMIW)MIW + \frac{1}{(NMOW)MOW}\right] \quad (2)$$

where

W	weight of a node
MWC	maximum weight among the node's children
F	factor of optimizing for load balancing versus clustering related objects
S	size of a node (object)
$NCID1$	number of children with in-degrees of 1
$NCIDM$	number of children with in-degrees greater than 1
SPC	size of all parent objects
MIW	maximum weight of incoming edges
$NMIW$	number of maximum-weighted incoming edges
MOW	maximum weight of outgoing edges
$NMOW$	number of maximum-weighted outgoing edges

The algorithm required to assign the weight of every node in the graph with time complexity of $O(n^2)$ (n is the number of nodes in the DAG) is as follows.

AssignWeights(DAG) Algorithm

1) for every node i (Starting at the leaf nodes and traversing the DAG in a breadth-first manner)
2) Calculate SPC_i
3) Calculate W_i

The CCP algorithm takes a DAG as its input and calls the AssignWeights Algorithm. The running time of the CCP algorithm is equal to the running time of AssignWeights plus the running time of the *repeat* loop. The loop has to be repeated n times and Line 4 can be done in $O(n)$. Therefore, the overall running time of the CCP algorithm is $O(n^2)$.

CCP(DAG) Algorithm

```
1) AssignWeights(DAG)
2) repeat until all the nodes have been processed
3)     Select the free node N with the largest weight
4)     if all parents of N are fully allocated on the channels
5)         place it on the currently least-loaded channel
6)     else
7)         Fill up the least-loaded channel(s) with nulls up to the end of the last
           allocated parent of N then place N on it.
```

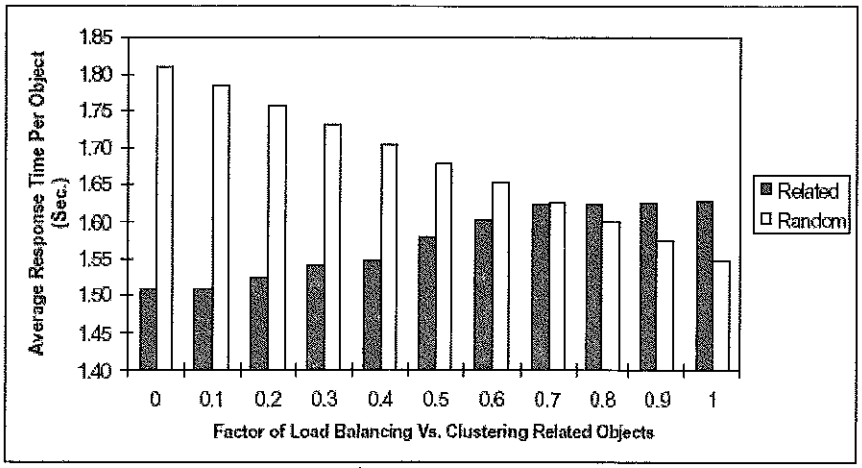
Performance Evaluation

To evaluate the performance of the proposed algorithms, our simulator was extended to measure the average response time per data item retrieval. To measure the effectiveness of the algorithms across a more unbiased test bed, the degree of connectivity among the data items in the DAG was randomly varied, and 100 different DAGs were generated. In every DAG, the out-degrees of the nodes were determined within the range of 0 and 3. To limit the experimentation running time, a decision was made to limit the number of nodes of each DAG to 60. The weights connecting the nodes, similar to the experiment reported in previously, were categorized as strong, normal, and weak, and were uniformly distributed along the edges of a DAG.

The simulation is accomplished in two steps: In the first step, every DAG is mapped onto the air channels using the LOF and CCP algorithms. In the second step, the simulator simulates the process of accessing the air channels in order to retrieve the data items requested in a query. Among the requested data items, 80% were selected based on their semantic relationship within the DAG and 20% were selected randomly. Finally, the average response time was calculated for 100,000 runs.

1) **Number of Air Channels.** As anticipated, increasing the number of channels resulted in a better response time for both the LOF and CCP. However, this improvement tapered off as the number of channels increased above a certain threshold value, since, additional parallelism provided by the number of channels did not match the number of free nodes available to be allocated, simultaneously. In addition, as expected, the CCP method outperformed the LOF method—the CCP heuristics attempt to smooth the distribution of the objects among the air channels while clustering the related objects.
2) **Out-Degree Distribution.** In general, the CCP method outperformed the LOF method. When the out-degree distribution is biased to include nodes with larger out-degrees (i.e., making the DAG denser), the LOF performance degrades at a much faster rate than the CCP method. This is due to the fact that such bias introduces more critical nodes and a larger number of children per node. The CCP method is implicitly capable of handling such cases.

Figure 10. Load balancing versus clustering



3) **Factor of Load Balancing versus Clustering Related Objects.** To get a better insight on the operations of the CCP method, we analyzed its behavior by varying the load balancing and degree of clustering (F; Equation 2). In this experiment, 80% of the data items requested by each query were related through certain semantic links and the rest were selected randomly. As can be seen (Figure 10), biasing in favor of clustering degrades the average response time for randomly selected data items. Optimization based on clustering increases the overall length of the broadcast, thus, contributing to larger response time for randomly accessed objects. For semantically related data items, however, decreasing F influenced the broadcast to favor the allocation of related data items closer to one another, thus improving the average response time. Such rate of improvement, however, declined as F reached a certain threshold value (0.2 in this case). At this point the behavior of the system reaches a steady state (the objects cannot be brought closer to one another). In different simulation runs, the ratio of randomly selected and semantically related data items varied in the ranges between 30/70% and 70/30% and the same behavior was observed. This figure can be productive in tuning the performance of the CCP method. Assuming a feedback channel is to be used to collect the statistics of the users' access pattern, F can be adjusted adaptively to match the access pattern. As an example, if the frequency of accessing data items based on their connection is equivalent to that of accessing data items randomly, then a factor value of 0.7 would generate the best overall response time.

Section Conclusion

This section concentrated on the proper mapping of data items on multiple parallel air channels. The goal was to find the most appropriate allocation scheme that would (a) preserve the connectivity among the data items, (b) provide the minimum overall broadcast time (load balancing), and (c) cluster related data items close to one another (improving the response time). Applying the LOF heuristic showed an improvement in load balancing. However, it proved short in solving the third aforementioned requirement. The CCP algorithm was presented to compensate this shortcoming. Relying on the critical path paradigm, the algorithm assumed several heuristics and showed better performance.

Energy-Efficient Indexing

In this section, we investigate and analyze the usage of indexing and indexed-based retrieval techniques for data items along the single and parallel broadcast channel(s) from an energy-efficient point of view. In general, index-based channel access protocols involve the following steps.

1) **Initial probe.** The client tunes into the broadcast channel to determine when the next index is broadcast.
2) **Search.** The client accesses the index and determines the offset for the requested data items.
3) **Retrieve.** The client tunes into the channel and pulls all the required data items.

In the initial probe, the mobile unit must be in active, operational mode. As soon as the mobile unit retrieves the offset of the next index, its operational mode could change to doze mode. To perform the *Search* step, the mobile unit must be in active mode, and when the unit gets the offset of the required data items, it could switch to doze mode. Finally, when the requested data items are being broadcast (*Retrieve* step), the mobile unit changes its operational mode to active mode and tunes into the channel to download the requested data. When the data is retrieved, the unit changes to doze mode again.

Figure 11. Inner-node structure of single-class and hierarchical schemes

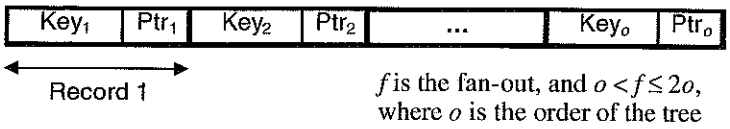
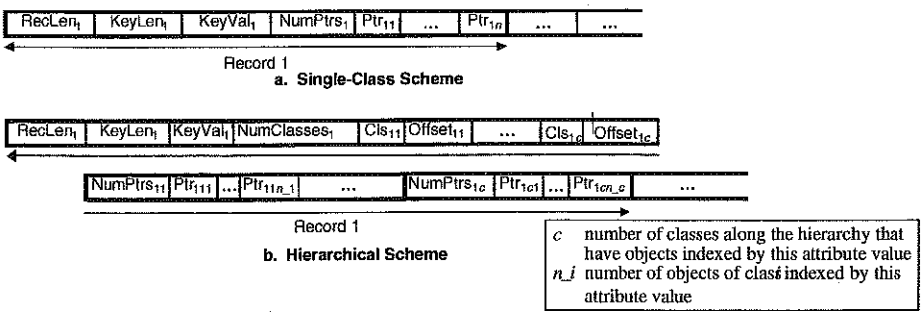


Figure 12. Leaf-node structure of single-class and hierarchical schemes



Object-Oriented Indexing

Object-oriented indexing is normally implemented as a multilevel tree. We can classify the possible implementation techniques into two general schemes: single-class indexing and hierarchical indexing. In the single-class scheme, multiple multilevel trees are constructed, each representing one class. In this case, the leaf nodes of each tree point to data items belonging only to the class indexed by that tree. A query requesting all objects with a certain ID has to navigate all these trees. On the other hand, the hierarchical scheme constructs one multilevel tree representing an index for all classes. The same query has to only navigate the common tree.

Data Indexing on a Single Air Channel

We assume an *air-channel page* as the storage granule on the air channel. Due to the sequential nature of the air channel, the allocation of the nodes of a multilevel tree has to follow the navigational path used to traverse the tree, starting at the root. Therefore, an ordering scheme is used to sequentially map the nodes on the air channel. Similarly, data items are allocated onto air channel pages following their index.

Storage Requirement

The overall storage requirement is the sum of the storage required by the inner and leaf nodes. For both schemes, the structure of the inner node is the same (Figure 11). An inner node is a collection of records, where each record is composed of a [*Key*, *pointer*] pair. Assume the order of the tree is o and the fan-out of every node is f ($o \leq f \leq 2o$, except for the root where $2 \leq f \leq 2o$). The leaf node structures of both schemes are shown in Figure 12. As can be seen, the main difference between the two schemes is that the hierarchical scheme requires a list of classes that have data items indexed by the index.

For the sake of simplicity, and without loss of generality, we assume that there are no overflow pages, furthermore assuming the following notations.

P	size of air-channel page
K	average number of distinct keys for an attribute
S	average size of a leaf-node index record in a single-class index
H	average size of a leaf-node index record in a hierarchical index
L	number of leaf-node pages
IN	number of inner-node pages for either scheme

$$L_{Single-Class} = \lceil K / \lfloor P / S \rfloor \rceil \quad (3)$$

$$L_{Hierarchical} = \lceil K / \lfloor P / H \rfloor \rceil \quad (4)$$

$$IN = 1 + \lfloor L / f \rfloor + \lfloor \lfloor L / f \rfloor / f \rfloor + \ldots \quad (5)$$

It should be noted that in the case of a single-class scheme, Equations 3 and 5 should be calculated for all the classes.

Timing Analysis

To perform the timing analysis, one has to consider the domain of a query. The cardinality of the domain of a query is the number of classes to be accessed by the query along the hierarchy. Our timing analysis evaluates the *response* and *active* time as the performance metrics. The response time is defined as the time elapsed between the first user access to the air channel and when the required information is retrieved. The active time is defined as the time during which the mobile unit has to be active accessing the channel. In the timing analysis, we use the number of pages as our unit of measurement. Finally, to support our protocols, we assume that every air-channel page contains control information indicating the location of the first page of the next index. This can simply be implemented as an offset (2 or 4 bytes).

a) **Hierarchical Method.** In this scheme, whether the domain of the query covers one class or all classes along the hierarchy, the same index structure has to be traversed. The protocol is shown below.

Hierarchical Protocol

1)	Probe onto channel and get offset to the next index	*active*
2)	Reach the index	*doze*
3)	Retrieve the required index pages	*active*

4) Reach the required data pages *doze*

5) Retrieve required data pages *active*

- **Response Time.** Assume I_H and D denote the size of the index and data, respectively. On average, it takes half the broadcast (the size of the broadcast is I + D) to locate the index from the initial probe. Once the index is reached, it has to be completely traversed before data pages appear on the broadcast. On average, it takes half the size of the data to locate and retrieve the required data items. Thus, the response time is proportional to:

$$\frac{I_H + D}{2} + I_H + \frac{D}{2} = \frac{3I_H}{2} + D = Broadcast + \frac{I_H}{2} \quad (6)$$

- **Active Time.** The mobile unit's modules have to be active to retrieve a page. Once the index is reached, a number of inner-node pages have to be accessed in order to get and retrieve a leaf-node page. The number of pages to be retrieved at the index is equal to the height of the index tree ($\log_f(D)$). Finally, the amount of the data pages to be read is equal to the number of data items to be retrieved that reside on distinct pages (NODP). Therefore, the active time is:

$$1 + \log_f(D) + NODP \quad (7)$$

b) **Single-Class Method.** In this scheme, we assume that the first page of every index contains information indicating the location of each index class. This structure can be implemented by including a vector of pairs [class_id, offset]. Assuming that the size of the offset and the class_id is 4 bytes each, the size of this structure would be 8c, where c is the number of class indexes on the broadcast.

Single-Class Protocol

1) Probe onto channel and get offset to the next index *active*

2) Reach the index *doze*

3) Retrieve offsets to the indexes of required classes *active*

4) for every required class

5) Reach the index *doze*

6) Retrieve the required index pages *active*

7) Reach the required data *doze*

8) Retrieve required data pages *active*

- **Response Time.** The size of a single index and its associated data are labeled as I_i and D_i, respectively. Since the total number of objects to be indexed is the same in single-class and hierarchical indexes, the sum of all D_i for all classes is equal to D. Assume a query references a set of classes where x and y stand for the first and last classes to be accessed. The average distance to be covered to get to x is half the distance covering the indexes and data between the beginning of y and the beginning of x. Once the index x is located, then all the indexes and data of all the classes between x and y (including those of x) have to be traversed. Once y is reached, its index and half of its data (on average) have to be traversed. Thus, the response time is proportional to:

$$\frac{\sum_{i=y}^{x-1}(I_i + D_i)}{2} + \sum_{i=x}^{y}(I_i + D_i) - \frac{D_y}{2} \tag{8}$$

Equation 8 provides a general means for calculating the average response time. However, the results are dependent on the location of the probe and the distance between x and y. It has been shown that the response time is lower bounded by half the size of the broadcast and upper bounded by slightly above the size of the broadcast. Further discussion on this issue is beyond the scope of this chapter and the interested reader is referred to Chehadeh, Hurson, and Kavehrad (1999).

- **Active Time.** Similar to the hierarchical case, the active time is dependent on the number of index pages and data pages to be retrieved. Therefore, the active time is the sum of the height of the trees for all the indexes of classes to be retrieved plus the number of the corresponding data pages. This is shown in the Equation 9. The 2 in the front accounts for the initial probe plus the additional page containing the index of classes (Line 3 in the protocol).

$$2 + \sum_{i=x}^{y}\left[\log_f(D_i) + NODP_i\right] \tag{9}$$

Performance Evaluation

Our simulator was extended to study both the response time and energy consumption with respect to the two allocation schemes. The overall structure of the schema graph determines the navigational paths among the classes within the graph. The relationships of the navigational paths within the graph influence the number and structure of indexes to be used.

- **Inheritance Relationship.** Within an inheritance hierarchy, classes at the lower level of the hierarchy inherit attributes of the classes at the upper level. Therefore,

Table 7. Input parameters

Parameter	Value (Default/Range)
Number of data items on Broadcast	5,120
Average Number of Classes Along Hierarchy	8
Percentage Distribution of Number of data items in Inheritance Hierarchy	25,25,25,25%
Percentage Distribution of Number of data items in Aggregation Hierarchy	40,30,20,10%
Distribution of data Size [S,M,L,VL]	16,512,3K,6K Bytes
Distribution of the data Sizes in Inheritance Hierarchy	VL,L,M,S
Distribution of the data Sizes in Aggregation Hierarchy	S,M,L,VL
Percentage of Classes to be Retrieved (Default/Range)	70% / [10-100%]
Average Number of data items to Retrieve Per Class	2
Fan-out in Index Tree	5
Average Number of data items with Distinct Key Attribute per Class	60% of data items per Class
Size of Air-Channel Page	512 Bytes
Broadcast Data Rate	1 M bits/sec
Power Consumption Active Mode	130 mW
Power Consumption Doze Mode	6.6 mW

Table 8. Number of index and data pages

	Aggregation/ Hierarchical	Aggregation/ Single [Eight Classes]	Inheritance/ Hierarchical	Inheritance/ Single [Eight Classes]
Index Pages	2,343	67,63,49,39,36,32,18,16	2,343	40,40,40,40,40,40,40,40
Data Pages	13,562	73,75,652,769,2504,28140,3206, 3502	34,015	12517,11520,5253,4252,637,440 ,25,20

data items belonging to the lower-level classes tend to be larger than those within the upper levels. The distribution of the number of data items is application dependent. In our analysis, and without loss of generality, we assumed the data items to be equally distributed among the classes of the hierarchy.

- **Aggregation Relationship.** In an aggregation hierarchy, data items belonging to lower classes are considered "part of" data items and those at the higher ends are the "collection" of such parts. Therefore, data items belonging to higher classes are generally larger than those belonging to the lower ones. In addition, the cardinality of a class at the upper end is smaller than a class at the lower end.

As a result, the organization of classes within the schema graph has its influence on the distribution of both the number and size of data items among the classes of the database. We assumed an average of eight classes for each hierarchy and categorize the sizes of data items as small, medium, large, and very large. Furthermore, 60% of the data items have distinct keys and the value of any attribute is uniformly distributed among the data items containing such attribute. Table 7 shows a list of all the input parameters assumed for this case.

Table 9. Response time degradation factor relative to the no-index scheme

Aggregation/ Hierarchical	Aggregation/ Single	Inheritance/ Hierarchical	Inheritance/ Single
1.17	1.05	1.1	1.02

The information along the broadcast channel is organized in four different fashions: the hierarchical and single-class methods for the inheritance and aggregation relationships. Table 8 shows the data and index page sizes for these organizations. Note that it is the number of data items (not data pages) that controls the number of index blocks. Within each indexing scheme, for each query, the simulator simulates the process of probing the air channel, getting the required index pages, and retrieving the required data pages. In each query, on average, two data items from each class are retrieved. The simulation measures the response time and amount of energy consumed.

a) **Response Time.** Placing an index along the air channel contributes to extra storage overhead and thus longer response time. Hence, the best response time is achieved when no index is placed, and the entire broadcast is searched. Table 9 shows the degradation factor in the average response time due to the inclusion of an index in the broadcast. The factor is proportional to the ratio of the size of the index blocks to that of the entire broadcast.

   Figure 13 shows the response time for all four different broadcast organizations. From the figure, one could conclude that for both the inheritance and aggregation cases, the response time of the hierarchical organization remained almost constant (with a slight increase, as the number of classes to retrieve increases). This is due to the fact that regardless of the number of classes and the location of the initial probe, all accesses have to be directed to the beginning of the index (at the beginning of the broadcast). The slight increase is attributed to the increase in the total number of objects to be retrieved — assuming that the objects to be retrieved

Figure 13. Response time versus number of retrieved classes

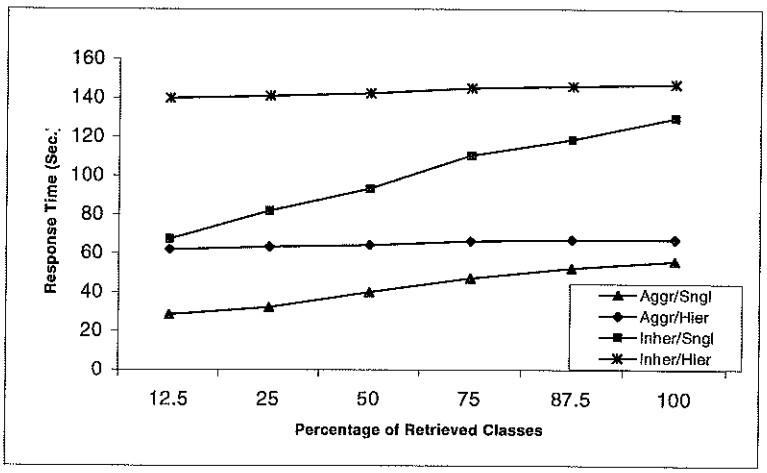
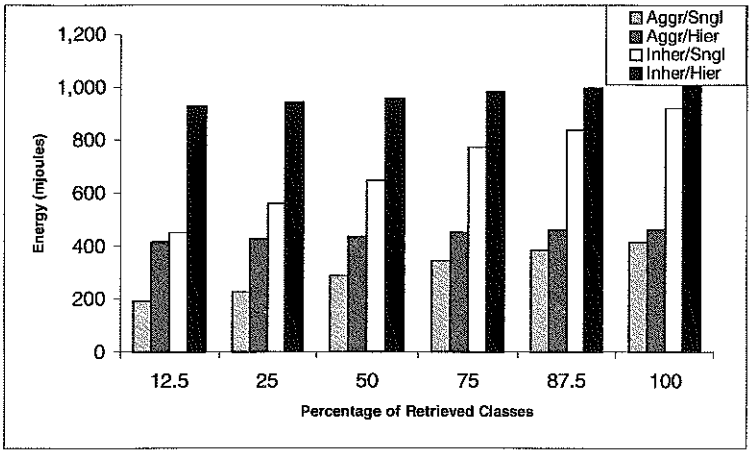


Figure 14. Detailed energy consumption



are distributed uniformly along the broadcast. It should be noted, however, that such an increase is only minor since the response time is mainly influenced by the initial procedure.

Two observations can be made: (a) the single-class method offers a better response time than the hierarchical case and (b) the response time for the single-class method increases as the number of retrieved classes increase. The first observation is due to the fact that in the single-class method, accesses do not have to be directed to the beginning of the broadcast. The second observation is due to the fact that an increase in the number of classes to be retrieved directly increases the number of index and data pages to be accessed.

Indexing based on the aggregation relationship offers lower response time than indexing based on the inheritance relationship since the distribution of the number of objects in the inheritance relationship is more concentrated on the larger objects. Having larger objects results in a longer broadcast, and, hence, it takes longer to retrieve the objects.

b) **Energy.** For each query, the amount of energy consumed is the sum of the energy consumed while the unit is in both active and doze modes. In the case where no index is provided, the mobile unit is in active mode during the entire probe. However, in the case where an index is provided, the active time is proportional to the number of index and data pages to be retrieved. As expected, the active time increases as the number of retrieved classes increases. The hierarchical method searches only one large index tree, whereas the single-class method searches through multiple smaller index trees. The number of pages to retrieve per index tree is proportional to the height of the tree. For a query spanning a single class, the single-class method produces a better active time than the hierarchical method. As the number of classes to be retrieved increases, the hierarchical tree is still traversed only once. However, more single-class trees have to be traversed, and, hence, results in an increase in the active time.

In both the single-class and the hierarchical methods, the aggregation case requires lower active time than the inheritance case since the inheritance case has larger objects, thus requiring the retrieval of more pages. For the sake of practicality, we utilized the power consumption data of the Hitachi SH7032 processor: 130 mW when active and 6.6 mW when in doze. Since power is the amount of energy consumed per unit of time, the total energy can be calculated directly using Equation 10:

$$Energy = (ResponseTime - ActiveTime)DozeModePower + (ActiveTime)ActiveModePower \quad (10)$$

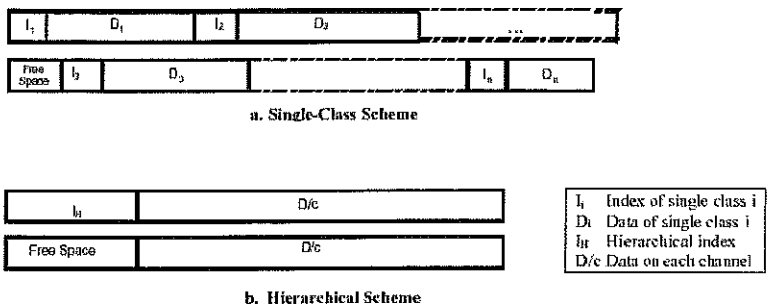
Figure 14 details the energy consumed during the entire query operation in mjoules. The power consumption of the mobile unit is much higher while the unit is in active mode, with a ratio of 19.7. However, our experiments showed that, in general, the duration of the active-mode operations was much smaller than doze-mode operations. As a result, the energy consumed during doze time was the dominating factor. As can be seen from Figure 14, the single-class method is superior to the hierarchical method. This is very similar to the results obtained for the response time, and similarly, the power consumption of the single-class method is lower than that of the hierarchical method.

Data Indexing on Parallel Air Channels

Allocation of Object-Oriented Indexing on Parallel Air Channels

Figure 15 shows the allocation of the single-class and hierarchical-based schemes on the two parallel air channels. For the single-class indexing scheme (Figure 15a), the index and data of each class are distributed and placed along the channels. The hierarchical indexing scheme (Figure 15b) places the index on one channel, and divides and distributes the data among the channels. The most popular data items can be put in the

Figure 15. Allocation of single-class and hierarchical indexes on two parallel air channels



a. Single-Class Scheme

b. Hierarchical Scheme

free space. Note that in both cases, similar to the single air channel, it is possible to interleave and distribute the index pages and associated data pages using a variety of methods.

Storage Requirement

In the case of broadcasting along parallel air channels, the storage requirement is the same as that for a single air channel.

Timing Analysis

In the case of parallel air channels, one has to account for switching between channels when analyzing access time and power consumption. During the switching time, the pages that are being broadcast on different channels cannot be accessed by the mobile unit. In addition, the mobile unit at each moment of time can tune into one channel—*overlapped page range*. By considering the average page size (512 bytes), communication bandwidth (1Mbit/sec), and switching time (the range of microseconds), we assume that the overlapped page range equals two pages. Finally, we assumed that the power consumption for switching between two channels is 10% of the power consumed in active mode. Equation 11 calculates the power consumption.

a) **Hierarchical Method.** The following protocol shows the sequence of operations.

Hierarchical Protocol

1)	Probe onto channel and retrieve offset to the next index	*active*
2)	Do {Reach the next index	*doze*
3)	Retrieve the required index pages	*active*
4)	Do {Reach the next possible required data page	*doze*
5)	Retrieve the next possible required data page	*active*
6)	}while every possible required data page is retrieved from the current broadcast	
7)	} while there are unaccessed data items because of overlapped page range	

$$EnergyConsumption = (ResponseTime - ActiveTime) * DozeModePower + (ActiveTime) * ActiveModePower + TheNumberOfSwitching * 10\% * ActiveModePower \quad (11)$$

- **Response Time.** I_H and D are used to denote the size of the hierarchy index and data, respectively. For the c-channel environment, the average size of data on each channel is D/c. To locate the index from the initial probe, it takes half the broadcast of one channel (the size of the broadcast is $I_H + D/c$). Once the index is reached, it has to be completely traversed before data pages appear on the broadcast and,

on average, it takes half the size of the data to locate and retrieve the required data items. Thus, the response time from the initial probe to the first complete broadcast is proportional to

$$(I_H + D/c)/2 + I_H + (D/c)/2 = 3I_H/2 + D/c \quad (12)$$

Because of the overlapped page range, the mobile units may not be able to get all of the required data during one complete broadcast (e.g., because of conflicts). Therefore, it has to scan the next broadcast. Let P be the probability of the data that are in the same overlapped page range. The distance from the last location to the next index is also half the size of the data of one channel. Once the index is reached, the same process will occur. Thus, the response time from the last location of the previous broadcast until the mobile unit can acquire all of the required data is proportional to

$$P * (D/2c + I_H + D/2c) = P * (I_H + D/c) \quad (13)$$

As a result, on average, the response time is proportional to

$$(1.5 + P) I_H + (1 + P) D/c \quad (14)$$

- **Active Time.** The mobile unit has to be active during the first probe (to retrieve a page). Once the index is reached, a number of nonleaf node pages have to be accessed in order to get and retrieve a leaf-node page. The number of pages to be retrieved at the index is equal to the height of the index tree ($\log_f(D)$). The amount of the data pages to be read is equal to the *NODP*. Again, because of the overlapped page range among parallel air channels, the probability of accessing the index of the next broadcast has to be included. Therefore, the active time is proportional to

$$1 + \log_f(D) + NODP + P * \log_f(D) \quad (15)$$

b) **Single-Class Indexing Scheme**

Single-Class Protocol

1)	Probe onto channel and retrieve offset to the next index	*active*
2)	Do {Reach the next index	*doze*
3)	Retrieve offsets to the indexes of required classes	*active*
4)	Reach the next possible index	*doze*
5)	Retrieve the next possible required index page	*active*

6) Do {Reach the next possible index or data page *doze*
7) Retrieve the next possible index or data page *active*
8) } while not (all indexes and data of required classes are scanned)
9) } while there are some data pages which are not retrieved because of overlapped page range

- **Response Time.** As before, the size of a single index and its associated data are labeled as I_i and D_i, respectively. The response time is simply driven by dividing Equation 8 by the number of the air channels (Equation 16):

$$\frac{\sum_{i=y}^{x-1}(I_i+D_i)}{2c}+\frac{\sum_{i=x}^{y}(I_i+D_i)}{c}-\frac{Dy}{2c} \tag{16}$$

Let P be the probability of the data that are in the same overlapped page range. Thus, the response time for getting the remaining required data items on the second broadcast probe is proportional to

$$P*(\frac{\sum_{i=y}^{x-1}(I_i+D_i)}{2c}+\frac{\sum_{i=x}^{y}(I_i+D_i)}{c}-\frac{D_y}{2c}) \tag{17}$$

As a result, the response time is proportional to

$$(1+P)*(\frac{\sum_{i=y}^{x-1}(I_i+D_i)}{2c}+\frac{\sum_{i=x}^{y}(I_i+D_i)}{c}-\frac{D_y}{2c}) \tag{18}$$

- **Active Time.** Similar to the hierarchical case, the active time is the sum of the height of the trees for all the indexes of the classes to be retrieved plus the number of the corresponding data pages. This is shown in Equation 19. The *2* at the beginning of the equation accounts for the initial probe plus the additional page containing the index of classes. Because of the overlapped page range among parallel air channels, the probability of accessing the index of the next broadcast has to be included. Therefore, the active time is proportional to

$$2+\sum_{i=x}^{y}\left[\log_f(D_i)+NODP_i\right]+\frac{P}{2}\sum_{i=x}^{y}\log_f(D_i) \tag{19}$$

Performance Evaluation

Once again, our simulator was extended to study the response time and energy consumption of the single-class and hierarchical indexing schemes in parallel air channels based on the input parameters presented in Table 7.

a) **Response Time.** In the case of no indexing, the response time was constant and independent of the number of channels. This is due to the fact that without any indexing mechanism in place, the mobile unit has to scan every data page in sequence until all required data pages are acquired. Moreover, when indexing schemes are in force, the response time lessens as the number of channels increases.

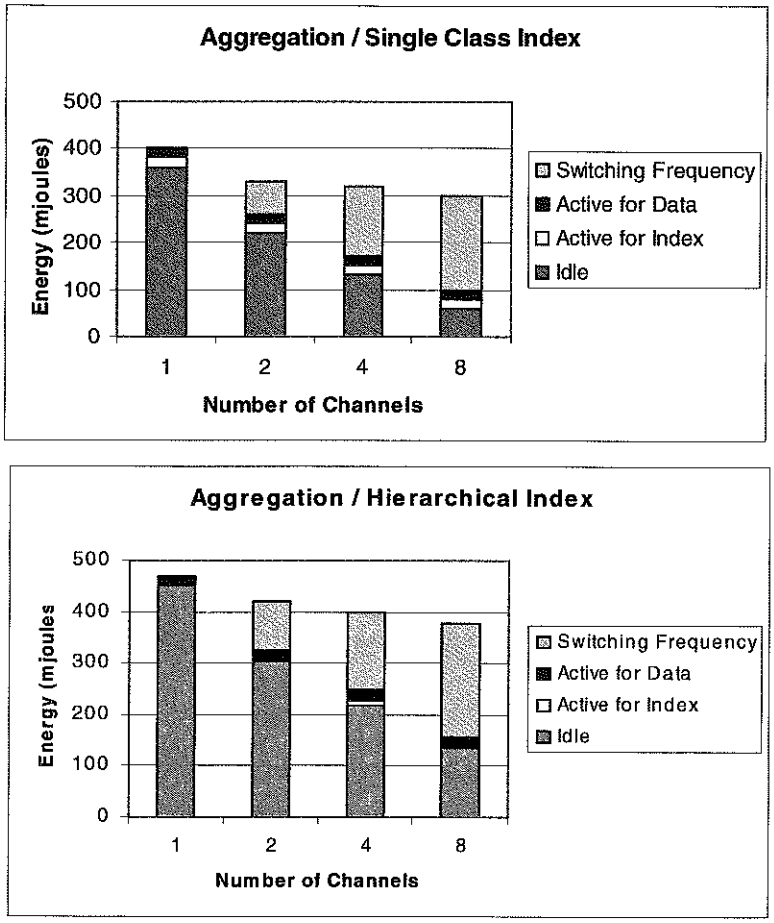
   For the inheritance and aggregation cases, the response time decreases as the number of channels increases. This is due to the fact that, as the number of channels increases, the length of the broadcast becomes shorter. However, the higher the number of channels, the higher the probability of conflicts in accessing data residing on different channels in the overlapped page range. As a result, doubling the number of channels will not decrease the response time by half.

   For both the inheritance and aggregation indexing schemes, the single-class method offers a better response time than the hierarchical method. The single-class method accesses do not have to be started at the beginning of the broadcast. For the hierarchical method, on the other hand, any access has to be started from the beginning of the broadcast, which makes the response time of the hierarchical method longer. Indexing based on the aggregation relationship offers a lower response time than that of the inheritance relationship because the distribution of data items in the inheritance relationship is more concentrated on the larger data items.

b) **Energy.** The active time is proportional to the number of index and data pages to be retrieved. For broadcast data without an index, the active time is the same as the response time. In addition, for all four indexing schemes, the active time remains almost constant and independent of the number of air channels. This is because the active time is proportional to the number of index and data pages to be retrieved.

   In general, the hierarchical method requires less active time than the single-class method. The hierarchical method searches only one large index tree, whereas the single-class method searches through multiple smaller index trees, and the number of pages to be retrieved per index tree is proportional to the height of the tree. In both the single-class and the hierarchical methods, the indexing based on an aggregation relationship requires lower active time than the inheritance method. This is simply due to the fact that the inheritance relationship resulted in larger data items, thus requiring the retrieval of more pages.

Figure 16. Detailed energy consumption



In a separate simulation run we observed the total energy consumption. It was concluded that the total energy consumption of broadcasting without any indexing schemes is much higher than that of broadcasting supported by indexing, and the energy consumption of the single-class method is lower than that of the hierarchical method. This is very similar to the results obtained for the response time. When indexing was supported, energy consumption, on average, decreased about 15 to 17 times in the case of the aggregation relationship and the inheritance relationship, respectively.

Figure 16 shows the detail of energy consumption for the aggregation relationship. As the number of channels increases, the energy consumption during idle time decreases. The energy consumption for retrieving indexes increases because the probability of the data being in the same overlapped page range increases. The

higher this probability, the more the mobile unit has to get the index from the next broadcast. Finally, the energy consumption for switching between two different channels increases because the required data are distributed among the channels. The larger the number of channels, the more distributed is the data among the channels, and, consequently, the more frequent switching between channels.

Section Conclusion

This section investigated an energy-efficient solution by the means of applying indexing schemes to object-oriented data broadcast over single and parallel air channels. Two methods, namely, the hierarchical and single-class methods, were explored. Timing analysis and simulation were conducted to compare and contrast the performance of different indexing schemes against each other. It was shown that including an index degrades the response time moderately, however, such degradation is greatly offset by the improvement in energy consumption. For a single air channel, broadcasting with supported indexing schemes increased the response time when compared with broadcasting without indexing support. However, the response time is reduced by broadcasting data with an index along parallel air channels. Moreover, the response time decreased as the number of air channels is increased. Relative to nonindexed broadcasting, the mobile unit's energy consumption decreased rather sharply when indexing is supported. For a set of queries retrieving data items along the air channel(s), the single-class indexing method resulted in a faster response time and lower energy consumption than the hierarchical method.

Conflicts and Generation of Access Patterns

One of the problems associated with broadcasting information on parallel air channels is the possibility for conflicts between accessing data items on different channels. Because the mobile unit can tune into only one channel at a time, some data items may have to be retrieved on subsequent broadcasts. In addition, during the channel switch time, the mobile unit is unable to retrieve any data from the broadcast. Conflicts will directly influence the access latency and, hence, the overall execution time. This section is intended to provide a mathematical foundation to calculate the expected number of passes required to retrieve a set of data items requested by an application from parallel air channels by formulating this problem as an *asymmetric traveling salesman problem* (TSP). In addition, in an attempt to reduce the access time and power consumption, we propose heuristic policies that can reduce the number of passes over parallel air channels. Analysis of the effectiveness of such policies is also the subject of this section.

Enumerating Conflicts

Equation 1 showed the number of broadcasts (passes) required to retrieve K data items from M parallel channels if conflicts between data items are independent. To calculate $P(i)$, it is necessary to count the number of ways the data items can be distributed while having exactly i conflicts, then divide it by the total number of ways the K data items can be distributed over the parallel channels. In order to enumerate possible conflicting cases, we classify the conflicts as single or double conflicts as defined below.

Definition 7. A single conflict is defined as a data item in the conflict region that does not have another data item in the conflict region in the same row. A double conflict is a data item that is in the conflict region and does have another data item in the conflict region in the same row.

The number of data items that cause a double conflict, d, can range from 0 (all single conflicts) up to the number of conflicts, i, or the number of remaining data items, $(K - i - 1)$. When counting combinations, each possible value of d must be considered separately. The number of possible combinations for each value of d is summed to determine the total number of combinations for the specified value of i. When counting the number of ways to have i conflicts and d double conflicts, four factors must be considered.

- Whether each of the $(i - d)$ data items representing a single conflict is in the left or right column in the conflict region. Because each data item has two possible positions, the number of variations due to this factor is $2^{(i-d)}$.
- Which of the $(M - 1)$ rows in the conflict region are occupied by the $(i - d)$ single conflicts. The number of variations due to this factor is $\binom{M-1}{i-d}$.
- Which of the $(M - 1) - (i - d)$ remaining rows in the conflict region are occupied by the d double conflicts; $(i - d)$ is subtracted because a double conflict cannot occupy the same row as a single conflict. The number of variations due to this factor is $\binom{(M-1)-(i-d)}{d}$.
- Which of the $(MN - 2M + 1)$ positions not in the conflict region are occupied by the $(K - i - d - 1)$ remaining data items. The number of variations due to this factor is $\binom{MN-2M+1}{K-i-d-1}$.

Note that these sources of variation are independent from each other and, hence:

$$P(i) = \frac{\sum_{d=0}^{d \le MIN(i, K-i-1)} 2^{(i-d)} \binom{M-1}{i-d} \binom{(M-1)-(i-d)}{d} \binom{MN-2M+1}{K-i-d-1}}{\binom{MN-1}{K-1}} \quad (20)$$

If the conflicts produced by one data item are independent from the conflicts produced by all other data items, then Equation 20 will give the number of passes required to retrieve all K requested data items. However, if the conflicts produced by one data item are not independent of the conflicts produced by other data items, additional conflicts will occur which are not accounted for in our analysis. Equation 20 will thus underestimate the number of broadcasts required to retrieve all K data items.

Retrieving Data from Parallel Broadcast Air Channels in the Presence of Conflicts

The problem of determining the proper order to retrieve the requested data items from the parallel channels can be modeled as a TSP. Making the transformation from a broadcast to the TSP requires the creation of a complete directed graph G with K nodes, where each node represents a requested object. The weight w of each edge (i, j) indicates the number of broadcasts that must pass in order to retrieve data item j immediately after retrieving data item i. Since any particular data item can be retrieved in either the current broadcast or the next broadcast, the weight of each edge will be either 0 or 1. A weight of 0 indicates that the data item j is after data item i in the broadcast with no conflict. A weight of 1 indicates that data item j is either before or in conflict with data item i.

Simulation Model

The simulation models a mobile unit retrieving data items from a broadcast. A broadcast is represented as an N x M two-dimensional array, where N represents the number of data items in each channel of a broadcast and M represents the number of parallel channels. For each value of K, where K represents the number of requested data items ($1 \leq K \leq NM$), the simulation randomly generates 1,000 patterns representing the uniform distribution of K data items among the broadcast channels. The K data items from each randomly generated pattern are retrieved using various retrieval algorithms. The number of passes is recorded and compared. To prevent the randomness of the broadcasts from affecting the comparison of the algorithms, the same broadcast is used for each algorithm in a particular trial and the mean value is reported for each value of K. Finally, several algorithms for ordering the retrieval from the broadcast, both TSP related and non-TSP related, were analyzed.

Data Retrieval Algorithms

Both exact and approximate TSP solution finders and two heuristic based methods were used to retrieve the data items from the broadcast.

a) **TSP Methods.** An exact TSP solution algorithm was used to provide a basis for comparison with the other algorithms. These algorithms are simply too slow and too resource intensive. While a better implementation of the algorithm may

somewhat reduce the cost, it cannot change the fact that finding the exact solution will require exponential time for some inputs. Knowing the exact solution to a given TSP does, however, allow us to evaluate the quality of a heuristic approach. A TSP heuristic based on the assignment problem relaxation requires far less CPU time and memory than the optimal tour finders, so it is suitable for use on a mobile unit. A publicly available TSP solving package named TspSolve (Hurwitz& Craig, 1996) was used for all TSP algorithm implementations.

b) **Next Data Item Access.** The strategy used by this heuristic is simply to always retrieve the next available data item in a broadcast. This can be considered as a greedy approach. It is also similar to the nearest neighbor approach to solving TSP problems.

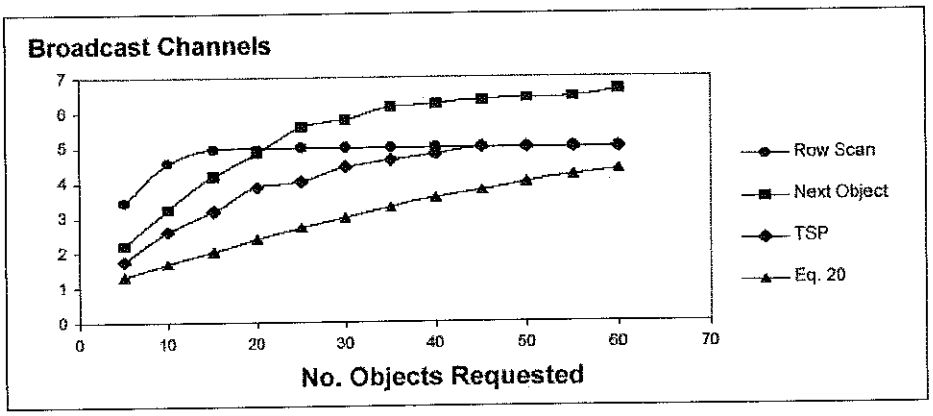
c) **Row Scan.** A simple *row scan* heuristic was also used. This algorithm simply reads all the data items from one channel in each pass. If a channel does not have any requested data in it, it is skipped. This algorithm will always require as many passes as there are channels with requested data items in them. The benefit of this algorithm is that it does not require any time to decide on an ordering. It can thus begin retrieving data items from a broadcast immediately. This is especially important when a large percentage of the data items in a broadcast are requested.

Results

As expected, the TSP methods provide much better results than both the two heuristic-based algorithms. Our simulations showed that the TSP heuristic performed almost exactly as well as the optimal TSP algorithm. This is a very interesting observation because it means that one can use a fast heuristic to schedule retrievals of data items from the broadcast without any performance degradation.

In Figure 17, the TSP methods show that the number of broadcasts required to retrieve all K requested data items from a broadcast is much greater than the number of broadcasts

Figure 17. Comparison of several algorithms for retrieving objects from parallel channels



predicted by Equation 20 — Equation 20 was based on the assumption that the conflicts among the requested data items are independent. Figure 17 used five parallel channels and 20 pages per channel. It is also interesting to note that the straightforward row scan nearly matches the performance of the TSP-based algorithms when more than about 45% of the total number of data items is requested. In this case, there are so many conflicts that it is virtually impossible to avoid having to make as many passes as there are parallel channels. When this occurs, it is better to do the straightforward row scan than to spend time and resources running a TSP heuristic.

Optimal Number of Broadcast Channels

More channels mean that a given amount of information can be made available in a shorter period of time at the expense of more conflicts. The simulation results showed that it is always advantageous to use more broadcast channels. While there will be more conflicts between data items, this does not quite counteract the shorter broadcast length of the many-channel broadcasts. This was especially evident when only a few data items in a broadcast were being accessed.

Ordered Access List

The scope of the general access protocol for indexed parallel-channel configuration in the presence of conflicts was extended in order to use heuristics that can generate the ordered access list of requested data items that reduces

- the number of passes over the air channels and
- the number of channel switches.

During the *Search* step, the index is accessed to determine the offset and the channel of the requested data items. Then, a sequence of access patterns is generated. Finally, the *Retrieval* step is performed following the generated access patterns.

Extended Retrieval Protocol

1) Probe the channel and retrieve the offset to the next index
2) Access the next index
3) Do {Search the index for the requested object
4) Calculate the offset of the object
5) Get the channel on which the object will be broadcast
6) } while there is an unprocessed requested object
7) Generate access patterns for the requested objects (using retrieval scheme)
8) Do {Wait for the next broadcast cycle

```
9)   Do {Reach the first object as indicated by the access pattern
10)     Retrieve the object
11)     } while there is an unretrieved object in the access pattern
12)     } while there is an unprocessed access pattern
```

Performance Evaluation

We extended the simulator to emulate the process of accessing data from a hierarchical indexing scheme in parallel air channels. Moreover, the simulator also analyzes the effect of conflicts on the average access time and power consumption.

Our retrieval scheme, based on the user request, generates a retrieval forest representing all possible retrieval sequences. However, as expected, the generated retrieval forest grows exponentially with the number of requested data items. The key observation needed to reduce the size of the tree is to recognize that each requested data item has a unique list of children, and the number of children for a particular data item is limited to the number of channels. The simulator takes advantage of these observations to reduce the size of the retrieval tree and the calculation time without sacrificing accuracy.

The generation of the user requests was performed randomly, representing a distribution of K data items in the broadcast. In various simulations runs, the value of K was varied from one to $N \times M$—in a typical user query of public data, K is much less than $N \times M$. Finally, to take into account future technological advances, parameters such as transmission rate and power consumption in different modes of operation were fed to the simulator as variable entities.

The simulator calculates the average active time, the average idle time, the average query response time, the average number of broadcast passes, the number of channel switches, and the energy consumption of the retrieval process. As a final note, the size of the index was 13.52% of the size of the broadcast (not including the index) and the number of channels varied from 1 to 16 (2 to 17 when an independent channel was used for transmitting the index).

Simulation Model

For each simulation run, a set of input parameters, including the number of parallel air channels, the broadcast transmission rate, and the power consumption in different

Table 10. Improvement of proposed algorithm versus row scan (10 data items requested)

# of Channels	# of Passes	Response Time	Energy
2	48.0%	28.0%	2.7%
4	68.0%	43.6%	3.1%
8	72.3%	46.5%	3.3%
16	71.8%	40.8%	3.4%

operational modes, was passed to the simulator. The simulator was run 1,000 times and the average of the designated performance metrics was calculated. The results of the simulations where an indexing scheme was employed were compared against a broadcast without any indexing mechanism. Two indexing scenarios were simulated.

- **Case 1.** The index was transmitted with the data in the first channel (index with data broadcast).
- **Case 2.** The index was transmitted over a dedicated channel in a cyclic manner.

Results

A comparison between the extended retrieval protocol against the row scan algorithm was performed. The index transmission was performed in a cyclic manner on an independent channel, and the number of requested data items was varied between 5 and 50 out of 5,464 securities within the NASDAQ exchange database. The simulation results showed that, regardless of the number of parallel air channels, the proposed algorithm reduces both the number of passes and the response time compared to the row scan algorithm. Moreover, the energy consumption was also reduced, but only when the number of data items retrieved was approximately 15 or less (Table 10).

Relative to the row scan algorithm, one should also consider the expected overhead of the proposed algorithm. The simulation results showed that in the worst case, the overhead of the proposed algorithm was slightly less than the time required to transmit one data page.

a) **Response Time.** Figures 18 to 20 show the response times in terms of the number of data items requested and the number of broadcast channels. Three cases were examined.

   - **Case 1. Data and index are intermixed on broadcast channel(s).** Figure 18 shows the response times for different numbers of broadcast channels when retrieving the full range of existing data items from the broadcast. It can be

Figure 18. Response time (Case 1)

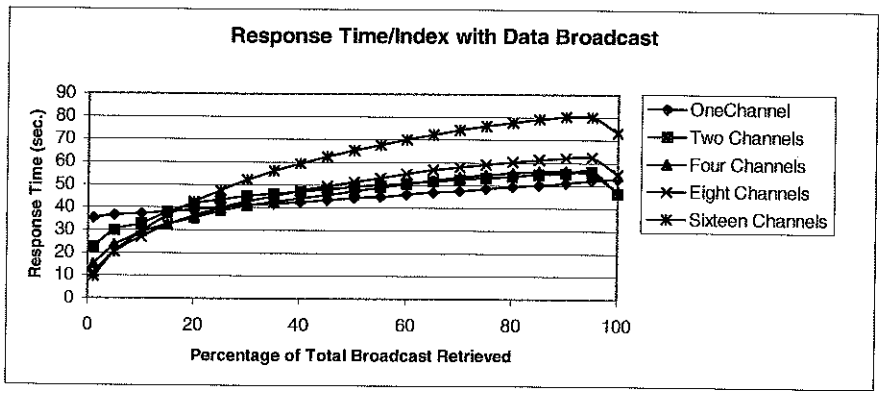
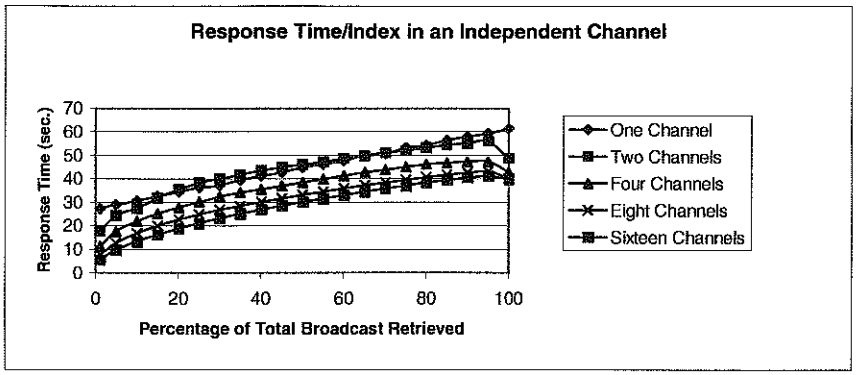


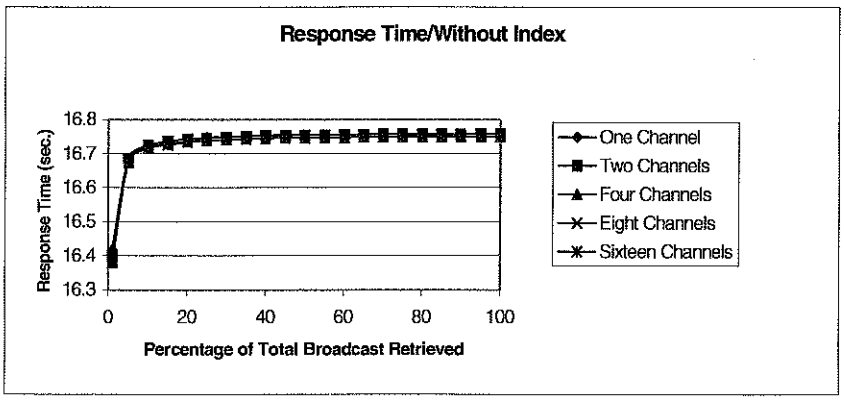
Figure 19. Response time (Case 2)



concluded that when a few data items are requested, the response time decreases as the number of channels increases. After a certain threshold point, the response time increases as the number of channels increases. This is due to an increase in the number of conflicts and hence an increase in the number of passes over the broadcast channels to retrieve the requested data.

- **Case 2. Index is broadcast over a dedicated channel in cyclic fashion.** Similar to Case 1, Figure 19 depicts the simulation results when retrieving the full range of existing data from the broadcast. Again, as expected, the response time decreases as the number of channels increases. Comparing the results for one-channel and two-channel configuration, we can conclude that in some instances the two-channel configuration is not as effective — there is the possibility of conflicts, many of which unavoidably cause an increase in the number of passes and hence longer response time.

Figure 20. Response time (Case 3)



- **Case 3. No indexing is employed.** From Figure 20 one can conclude that the response time remains relatively constant regardless of the number of channels used. In this organization, the user must scan the same amount of data regardless of the user query and number of parallel channels.

In general, employment of an indexing scheme reduces the response time when retrieving a relatively small number of data items. As the percentage of data items requested increases, the number of conflicts increases as well. The proposed retrieval protocol tries primarily to reduce the conflicts in each pass of the broadcast; however, when the number of potential conflicts increases considerably, some conflicts become unavoidable, causing an increase in the number of passes and hence an increase in the response time. When the percentage of requested data approaches 100%, the response time reduces. This proves the validity of the proposed scheduling algorithm since it generates the same retrieval sequence as the row scan method.

b) **Switching Frequency.** Again, three cases were examined.

- **Case 1 & Case 2. Employment of indexing schemes.** Figure 21 shows the switching frequency for Case 1 and Case 2—the switching pattern is not affected by the indexing policy employed. From this figure one can conclude that the switching frequency increases as the number of channels and number of data items retrieved increase. This can be explained by an increase in the number of conflicts; as the proposed method tries to reduce the number of conflicts, the switching frequency will increase. Also, as stated previously, an increase in the number of channels increases the number of conflicts as well. One can notice that when the percentage of data items requested exceeds 50%, the switching frequency begins to decrease. This is due to the fact that the proposed method does not attempt to switch channels as often to avoid the conflicts as the number of conflicts increases substantially.
- **Case 3. No indexing is employed.** When no indexing technique is utilized, the row scan method is employed, producing a constant switching frequency

Figure 21. Switching frequency (Case 1 and Case 2)

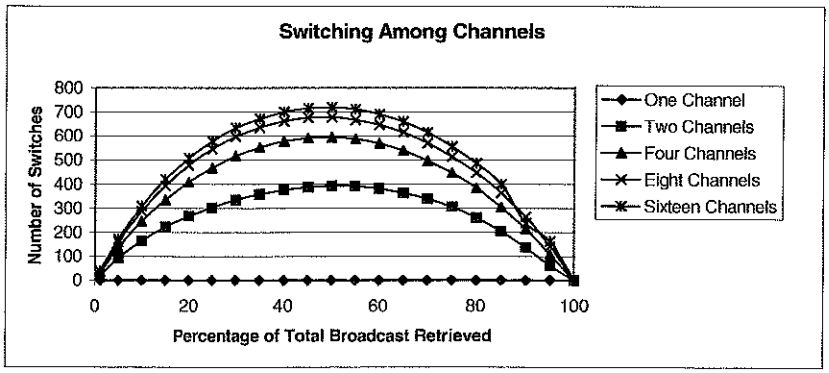


Figure 22. Energy consumption (Case 1)

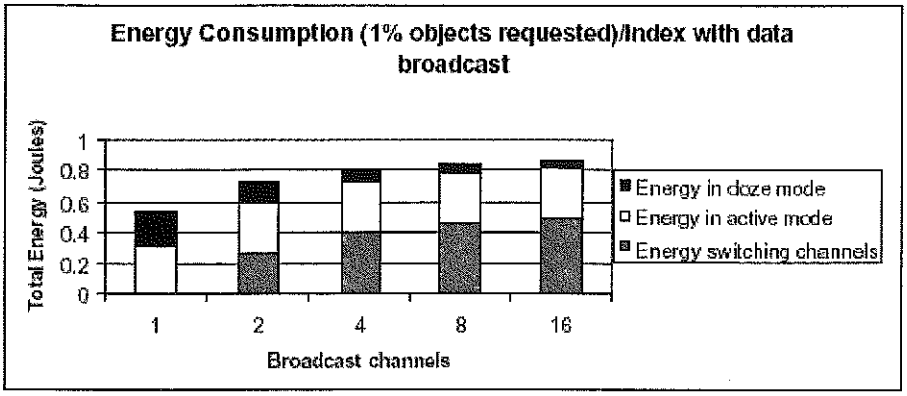


Figure 23. Energy consumption (Case 2)

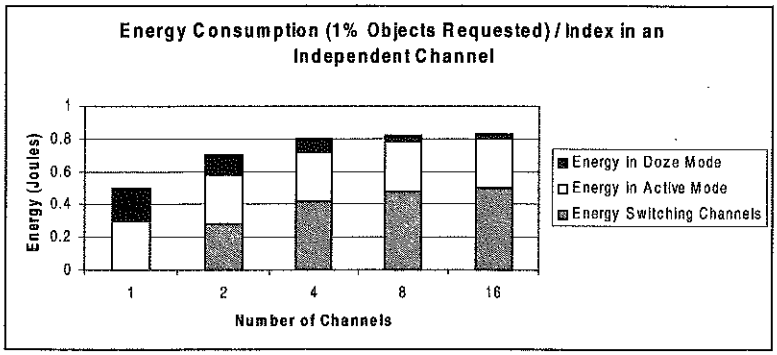


Figure 24. Energy consumption (Case 3)

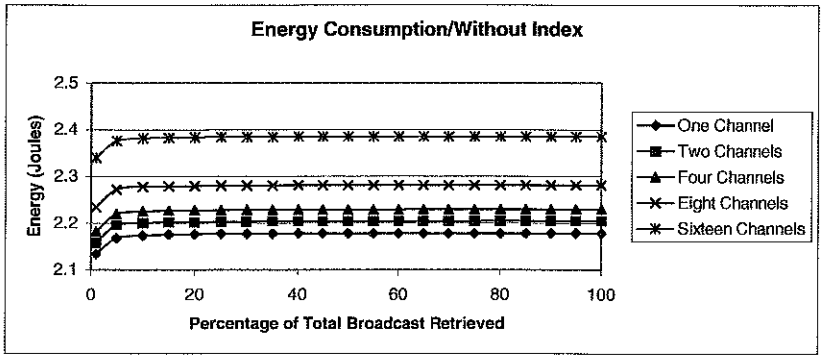
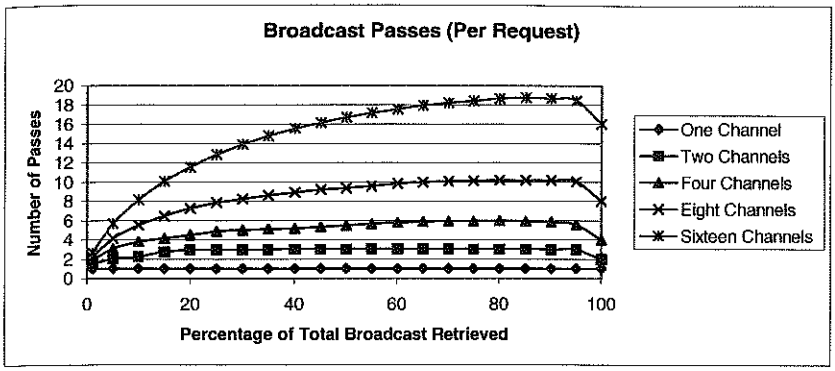


Figure 25. Number of passes over the parallel channels



independent of the number of data items requested. The switching frequency is, at the most, equal to the number of total channels employed in the simulation.

c) **Energy Consumption.** Figures 22 and 23 depict detailed energy consumption when 1% of data items on the broadcast are requested. It can be observed that the energy consumption is almost the same; however, Case 1 consumes more energy than Case 2 in doze mode.

- **Case 1 & Case 2. Employment of indexing schemes.** In general, due to the increase in the number of channel-switching frequency, the energy consumption increases as the number of channels increases. In addition, we noted that the energy consumption increases when up to 50% of the broadcast data items are requested, then it decreases as the number of requested data items increases. This is directly related to the channel-switching frequency. In Case 1, in many instances, the mobile unit must wait in doze mode while the index is retransmitted.
- **Case 3. No indexing is employed.** When no indexing technique is used (Figure 24), the energy consumption varies only minimally due to the nature of the row scan algorithm employed.

From these figures we can observe that both Case 1 and Case 2 consume less power than Case 3 when a small percentage of data items is retrieved (around 1%). When the percentage of data items requested increases, the number of conflicts, the switching frequency, and, consequently, the energy consumption increase.

d) **Number of Passes.** As a note, the number of passes is independent of the index allocation scheme. Therefore, the number of passes for Cases 1 and 2 is the same.

- **Case 1 & Case 2. Employment of indexing schemes.** The increase in the number of passes is directly related to the increase in the number of channels and increase in the number of data items requested (Figure 25). An increase

in the number of channels implies an increase in the number of conflicts, and, hence, the higher possibility of unavoidable conflicts, resulting in an increase in the number of passes. It can be noticed that when the number of data items requested is large, the number of passes exceeds the number of channels available. This is due to the priority order of the heuristics used in the proposed retrieval algorithm. In general, it is improbable that a query for public data requests a lot of data items from the broadcast channels. Our experience showed that for a query requesting up to 50 data items, the proposed method reduces the number of passes compared to Case 3.

- **Case 3. No indexing is employed.** In contrast, when no indexing technique is employed, the number of passes required is a function of the number of air channels. In the worst case we need N passes where N is the number of broadcast channels.

Section Conclusion

Conflicts directly influence the access latency and, hence, the overall execution time. This section provided a mathematical foundation to calculate the expected number of passes required to retrieve a set of data items requested by an application from parallel air channels. In addition, in an attempt to reduce the access time and power consumption, heuristics were used to develop access policies that reduce the number of passes over the parallel air channels. Analysis of the effectiveness of such policies was also the subject of this section.

Conclusion and Future Research Directions

This chapter aims to address the applicability and effectiveness of data broadcasting from two viewpoints: energy and response time. Within the scope of data broadcasting, we discussed different data allocation schemes, indexing approaches, and data retrieval methods for both single and parallel air channels. Comparisons of different algorithms were demonstrated through simulation results.

The scope of this research can be extended in many directions. For instance, we assumed that the resolution of queries happens on an individual basis at the mobile unit. It may be possible to reduce computation by utilizing a buffer and bundling several queries together, processing them as a whole. Our proposed scheduling scheme was based on three prioritized heuristics. It is interesting to investigate a new set of heuristics that can reduce the switching frequency while retrieving a large percentage of data items from the broadcast.

Acknowledgment

This work would have not been possible without the sincere effort of many students who participated in the development of conceptual issues as well as simulation results. We would like to thank them. In addition, this work in part has been supported by the Office of Naval Research and the National Science Foundation under the contracts N00014-02-1-0282 and IIS-0324835, respectively.

References

Acharya, S., Alonso, R., Franklin, M., & Zdonik, S. (1995). Broadcast disks: Data management for asymmetric communication environments. *Proceedings of ACM SIGMOD International Conference on the Management of Data*, (pp. 199-210).

Alonso, R., & Ganguly, S. (1992). Energy efficient query optimization. *Technical Report MITL-TR-33-92*, Princeton, NJ: Matsushita Information Technology Laboratory.

Alonso, R., & Korth, H. F. (1993). Database system issues in nomadic computing. *Proceedings of ACM SIGMOD Conference on Management of Data*, (pp. 388-392).

Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D., & Zdonik, S. (1989). The object-oriented database system manifesto. *Proceedings of Conference on Deductive and Object-Oriented Databases*, (pp. 40-57).

Badrinath, B. R. (1996). Designing distributed algorithms for mobile computing networks. *Computer Communications*, *19*(4), 309-320.

Banerjee, J., Kim, W., Kim, S.-J., & Garza, J. F. (1988). Clustering a DAG for CAD databases. *IEEE Transactions on Software Engineering, 14*(11), 1684-1699.

Boonsiriwattanakul, S., Hurson, A. R., Vijaykrishnan, N., & Chehadeh, C. (1999). Energy-efficient indexing on parallel air channels in a mobile database access system. *Proceedings of the Third World Multiconference on Systemics, Cybernetics, and Informatics, and Fifth International Conference on Information Systems Analysis and Synthesis, IV,* (pp. 30-38).

Bowen, T. F. (1992). The DATACYCLE architecture. *Communication of ACM, 35*(12), 71-81.

Bright, M. W., Hurson, A. R., & Pakzad, S. (1992). A taxonomy and current issues in multidatabase systems. *IEEE Computer*, *25*(3), 50-60.

Bright, M. W., Hurson, A. R., & Pakzad, S. (1994). Automated resolution of semantic heterogeneity in multidatabases. *ACM Transactions on Database Systems*, *19*(2), 212-253.

Chang, E. E., & Katz, R. H. (1989). Exploiting inheritance and structure semantics for effective clustering and buffering in an object-oriented DBMS. *Proceedings of ACM SIGMOD Conference on Management of Data*, (pp. 348-357).

Chehadeh, Y. C., Hurson, A. R., & Tavangarian, D. (2001). Object organization on single and parallel broadcast channel. *Proceedings of High Performance Computing*, (pp. 163-169).

Chehadeh, Y. C., Hurson, A. R., & Kavehrad, M. (1999). Object organization on a single broadcast channel in the mobile computing environment [Special issue]. *Multimedia Tools and Applications Journal, 9*, 69-94.

Chehadeh, Y. C., Hurson, A. R., & Miller L. L. (2000). Energy-efficient indexing on a broadcast channel in a mobile database access system. *Proceedings of IEEE Conference on Information Technology*, (pp. 368-374).

Chehadeh, Y. C., Hurson, A. R., Miller, L. L., Pakzad, S., & Jamoussi, B. N. (1993). Application of parallel disks for efficient handling of object-oriented databases. *Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Processing*, (pp. 184-191).

Cheng, J.-B. R., & Hurson, A. R. (1991a). Effective clustering of complex objects in object-oriented databases. *Proceedings of ACM SIGMOD Conference on Management of Data*, (pp. 22-27).

Cheng, J.-B. R., & Hurson, A. R. (1991b). On the Performance issues of object-based buffering. *Proceedings of International Conference on Parallel and Distributed Information Systems*, (pp. 30-37).

Chlamtac, I., & Lin, Y.-B. (1997). Mobile computing: When mobility meets computation. *IEEE Transactions on Computers, 46*(3), 257-259.

Comer, D. C. (1991). *Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.

Demers, A., Pertersen, K., Spreitzer, M., Terry, D., Theier, M., & Welch, B. (1994). The bayou architecture: Support for data sharing among mobile users. *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, (pp. 2-7).

Fong, E., Kent, W., Moore, K., & Thompson, C. (1991). *X3/SPARC/DBSSG/OODBTG Final Report*. Available from NIST.

Fox, A., Gribble, S. D., Brewer, E. A., & Amir, E. (1996). Adapting to network and client variability via on-demand dynamic distillation. *Proceedings of ASPLOS-VII*, Boston, Massachusetts, (pp. 160-170).

Honeyman, P., Huston, L., Rees, J., & Bachmann, D. (1992). The LITTLE WORK project. *Proceedings of the Third IEEE Workshop on Workstation Operating Systems*, (pp. 11-14).

Hu, Q.L., & Lee, D. L. (2000). Power conservative multi-attribute queries on data broadcast. *Proceedings of IEEE International Conference on Data Engineering (ICDE 2000)*, (pp. 157-166).

Hu, Q. L., & Lee, D. L. (2001). A hybrid index technique for power efficient data broadcast. *Distributed and Parallel Databases Journal, 9*(2), 151-177.

Hurson, A. R., Chehadeh, Y. C., & Hannan, J. (2000). Object organization on parallel broadcast channels in a global information sharing environment. *Proceedings of IEEE Conference on Performance, Computing, and Communications*, (pp. 347-353).

Hurson, A. R., Pakzad, S., & Cheng, J.-B. R. (1993). Object-oriented database management systems. *IEEE Computer, 26*(2), 48-60.

Hurwitz, C. & Craig, R. J. (1996). *Software Package Tsp_Solve 1.3.6.* Available from http://www.cs.sunysb.edu/~algorithm/implement/tsp/implement.shtml.

Imielinski, T., & Badrinath, B. R. (1994). Mobile wireless computing: Challenges in data management. *Communications of the ACM, 37*(10), 18-28.

Imielinski, T., & Korth, H. F. (1996). Introduction to mobile computing. In T. Imielinski and H. F. Korth (Eds.), *Mobile computing* (pp. 1-43). Boston: Kluwer Academic.

Imielinski, T., Viswanathan, S., & Badrinath, B. R. (1994). Energy efficient indexing on air. *Proceedings of ACM SIGMOD Conference on Management of Data*, (pp. 25-36).

Imielinski, T., Viswanathan, S., & Badrinath, B. R. (1997). Data on air: Organization and access. *IEEE Transactions on Computer, 9*(3), 353-372.

Joseph, A. D., Tauber, J. A., & Kaashoek, M. F. (1997). Mobile computing with the rover toolkit [Special issue]. *IEEE Transactions on Computers, 46*(3), 337-352.

Juran, J., Hurson, A. R., & Vijaykrishnan, N. (2004). Data organization and retrieval on parallel air channels: Performance and energy issues. *ACM Journal of WINET, 10*(2), 183-195.

Kaashoek, M. F., Pinckney, T., & Tauber, J. A. (1994). Dynamic documents: Mobile wireless access to the WWW. *IEEE Workshop on Mobile Computing Systems and Applications*, 179-184.

Kim, W. (1990). *Introduction to object-oriented databases.* Cambridge, MA: MIT Press.

Lai, S. J., Zaslavsky, A. Z., Martin, G. P., & Yeo, L. H. (1995). Cost efficient adaptive protocol with buffering for advanced mobile database applications. *Proceedings of the Fourth International Conference on Database Systems for Advanced Applications.*

Lee, D. L. (1996). Using signatures techniques for information filtering in wireless and mobile environments [Special issue]. *Distributed and Parallel Databases, 4*(3), 205-227.

Lee, M. T., Burghardt, F., Seshan, S., & Rabaey, J. (1995). InfoNet: The networking infrastructure of InfoPad. *Proceedings of Compcon*, (pp. 779-784).

Lim, J.B., & Hurson, A. R. (2002). Transaction processing in mobile, heterogeneous database systems. *IEEE Transactions on Knowledge and Data Engineering, 14*(6), 1330-1346.

Lim, J. B., Hurson, A. R., Miller, L. L., & Chehadeh, Y. C. (1997). A dynamic clustering scheme for distributed object-oriented databases. *Mathematical Modeling and Scientific Computing*, 8, 126-135.

Munoz-Avila, A., & Hurson, A. R. (2003a). Energy-aware retrieval from indexed broadcast parallel channels. *Proceedings of Advanced Simulation Technology Conference (High Performance Computing)*, (pp. 3-8).

Munoz-Avila, A., & Hurson, A. R. (2003b). Energy-efficient objects retrieval on indexed broadcast parallel channels. *Proceedings of International Conference on Information Resource Management*, (pp. 190-194).

NASDAQ World Wide Web Home Page. (2002). Retrieved May 11, 2004, from http://www.nasdaq.com

Satyanarayanan, M. (1996). Fundamental challenges in mobile computing. *Proceedings of 15th ACM Symposium on Principles of Distributed Computing*, (pp. 1-7).

Satyanarayanan, M., Noble, B., Kumar, P., & Price, M. (1994). Application-aware adaptation for mobile computing. *Proceedings of the Sixth ACM SIGOPS European Workshop,* (pp. 1-4).

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM, 36*(7), 75-84.

Zdonik, S., Alonso, R., Franklin, M., & Acharya, S. (1994). Are disks in the air just pie in the sky? *Proceedings of Workshop on Mobile Computing Systems and Applications*, (pp. 1-8).