**DIMACS Technical Report 95-11**
**May 1995**

# Airdisks and AirRAID: Modeling and scheduling periodic wireless data broadcast (Extended Abstract)[1]

by

*Ravi Jain*[2,3]
Applied Research
Bellcore

*John Werth*
Dept of Computer Sciences
Univ. of Texas at Austin

[1]An earlier version of this paper was accepted for presentation at *5th WINLAB Workshop on Third Generation Wireless Information Networks*, East Brunswick, NJ, Apr. 26-27, 1995.
[2]Permanent Member
[3]Address correspondence to Ravi Jain, Applied Research, Bellcore, 445 South St, Morristown, NJ 07960. e-mail: `rjain@thumper.bellcore.com`

---

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

**ABSTRACT**

We introduce a simple model, called the airdisk, for modeling the access of data transmitted periodically over wireless media as being analogous to the access of data from a standard magnetic disk. We consider several issues related to airdisks, such as their mean rotational latency under certain assumptions. The problem of scheduling the order in which data items are broadcast is analogous to that of determining how data should be laid out on the disk. Two problems of laying out data so as to minimize read time, given information about which data items are of most interest to the clients, are defined; both are shown to be NP-complete. We discuss ways in which the information about which items are of interest to clients can be obtained. Finally we consider how to increase the performance and storage capacity of airdisks, using the magnetic disk analogy as a guide. We suggest using multiple-track airdisks or borrowing the idea of Redundant Arrays of Inexpensive Disks (RAID) which is used for magnetic disks; for the wireless data broadcast environment we call the latter approach airRAID.

# 1 Introduction

The delivery of data over wireless media is rapidly becoming an important and expanding application area. It seems likely that the use of various wireless media, such as paging, FM subcarrier, cellular data, and PCS wireless networks [3], will continue to increase for delivery of a wide variety of data applications. Applications in the realm of personal information services, such as the delivery of periodically updated news, weather, stock, traffic, and sports information to mobile users, appear to be poised for explosive growth and widespread proliferation. We focus on this class of applications in this paper.

The constraints imposed by wireless communication and user mobility, and the resulting limitations in user terminal capabilities, pose interesting challenges in the design of schemes for efficient delivery of personal information services. A significant amount of attention has been paid to the development of wireless data protocols for the various wireless communication media mentioned above. Recently, Imielinski et al [14] have considered the problem of retrieving data using an index, as is commonly done for magnetic disks, where both the data and the index are broadcast periodically over a wireless medium. This approach builds upon previous work done at Bellcore on the Datacycle computer architecture [9, 7], in which data is periodically broadcast over a fixed high-bandwidth *wireline* network to receivers capable of filtering the information relevant to them at high speeds.

We take the ideas of Datacycle and Imielinski et al a small step further by modeling the periodic broadcast of data as a virtual disk, which we call an *airdisk*. (In the rest of this paper, the term "disk" or "airdisk" is used to refer to a virtual disk, while a standard magnetic disk will be explicitly called a "magnetic disk".) The performance characteristics of the periodic data broadcasts can then be cast in the same terms as those used for magnetic disks, such as rotational latency, seek time and transfer rate. In sec. 2 we describe the airdisk model, where a (logically) centralized server broadcasts data (writes on the disk) and many clients can receive the broadcast (read the disk) and also send messages to the server to modify the content of the next broadcast (i.e., write the disk).

In sec. 3 we consider the case where the airdisk is a One-Writer Many-Reader (OWMR) disk. In this case only the server can write the disk. We show expressions for the maximum and mean rotational latency of such a disk under certain assumptions.

One of the significant advantages of an airdisk compared to a magnetic disk is that the layout of the data on the disk can be changed easily in response to changes in the data access patterns. For magnetic disks such operations are sometimes carried out in order to overcome the I/O performance bottleneck presented by magnetic disks in parallel computing. For example, del Rosario et al [6] describe compiler techniques to detect different data access patterns in different phases of a parallel program; they then show how changing the data layout between two phases of a parallel program can significantly improve overall performance. However, such operations are expensive for magnetic disks as they involve reading, buffering, and re-writing of data. For airdisks, on the other hand, where data is broadcast afresh at every period, they involve negligible overhead at the sender and can be of substantial benefit in improving performance and conserving the limited resources of mobile clients.

In sec. 4 we study how the data layout on an airdisk could be changed, if information about data access patterns is available, so as to minimize mean access time. The data layout for a given broadcast period is determined simply by the order in which data items are broadcast by the sender. We use airdisk access time as a measure of performance. (We also briefly discuss issues of energy consumption at the mobile client.) We model two problems of determining the optimal broadcast scheduling (i.e., airdisk data layout) in graph-theoretical terms and show that they are NP-complete [10], i.e., computationally intractable. We are currently investigating sub-optimal heuristics for these problems. In sec. 5 we consider the problem of obtaining information about data access patterns in a wireless mobile environment, and discuss several alternative solutions.

One of the problems with airdisks is that as the quantity of data to be broadcast increases, the delays involved in accessing data increase. We use the airdisk model to borrow the solution to this problem used for magnetic disks, i.e., to improve performance via parallelism while maintaining availability via redundancy, as in the Redundant Arrays of Inexpensive Disks (RAID) approach [13, 11] which is in successful and widespread use. We call this design the *AirRAID*, and we describe it very briefly in sec. 6.

# 2 Disks and airdisks

We first summarize magnetic disk terminology [11]. A magnetic disk consists of a set of circular *platters*, each of which rotates under a read/write *head* which can move radially across the platter surface. Typically, only a single platter can be read or written at a time. The platter is divided radially into *sectors*. An annular portion of data on the platter, which can be accessed by moving the head only once to the appropriate radial position, is called a *track*. The time to access data on the magnetic disk can be roughly divided into three components. The *seek time* is the time to move the head from its current position to the appropriate track. The *rotational latency* is the time which the head spends waiting for the platter to rotate so that the desired sector is accessible. The *transfer time* is the time required by the head to then physically read or write the data.

## 2.1 System model

We consider the delivery of personal information services via wireless broadcast described as follows. An information services provider generates data. A fixed, (logically) centralized information server periodically broadcasts the data to a large number of mobile clients via a wireless medium. As an example, the server receives updates of current stock prices from the stock exchange, updates its database, and periodically broadcasts the updated information to the clients. The clients receive the broadcasts and filter out the information which is not desired. A client typically consists of a battery-powered portable device, such as a palmtop Personal Digital Assistant (PDA) or a laptop computer. (Note that a client is a device, while the ultimate human recipient of the data is called the user.)

In the simplest case, data transmission is simplex (one-way), from the server to the clients. Communication could also be half- or full-duplex. The communication channel from the server to the clients is called the *downlink*, and from the clients to the server is called the *uplink*. If an uplink channel exists it is shared amongst the clients by means of ALOHA or Ethernet-type contention protocols, or using TDMA, FDMA, or CDMA multiplexing [15]. The clients can use the uplink channel to send read or write requests to the server. The server responds to read and write requests by modifying the stream of data items sent in subsequent broadcast periods. In this paper we will assume that the uplink, if any, is not used for sending acknowledgements to the server about which data items have been correctly received, since the data items sent by the server are not addressed to individual clients and, depending upon radio channel conditions, some clients may receive them correctly and some may not. Thus the server cannot know which clients received the information correctly and which did not. The server inserts forward error correction [15] encoding into the data written onto the airdisk, and typically periodically re-broadcasts the data even if it has not been updated by the information provider.

The architecture and details of the communications system depend upon the technology used. If paging is used, the wireless bandwidth is low and (until very recently) limited to simplex communication. FM subcarrier communication can potentially provide higher bandwidth and full-duplex communication, although currently most systems are used in low-bandwidth simplex mode. Cellular data communication typically provides full-duplex communication, with a low to moderate bandwidth. PCS systems such as Bellcore's WACS [4] typically provide full-duplex services and data rates up to around 32 kbps. We omit further details of the architecture of the server and of the communication network.

Each broadcast period consists of a *preamble*, a pattern of zero or more bits known to the clients and used for synchronization. The preamble is followed by a broadcast *period flag*, a pattern of bits indicating the start of the data in the period. In general, the period flag is followed by an *index* which gives the sequence of data items broadcast in this period. Each data item has an *item header* at the start and a *trailer* indicating its end. Typically, the item header identifies the item (e.g., it specifies that the item contains current market information about gold prices) and may contain other information also (e.g. length of the item), while the trailer is simply a known pattern of bits marking the end of the item.

Depending upon the scheme used for sending the data, the preamble, index, headers and trailers may be of zero or more bits. The item headers and trailers are not required if all items are of the same length, and that length is fixed and known to the clients. An index is not required if the data items are fixed and sent in a fixed sequence known to the clients.

In this paper we will consider situations where the length and sequence of data items is fixed, as well as those where it is not.

# 3 Simplex broadcast communication

In this section we consider the situation in which communication is simplex and the clients simply filter the data broadcast by the server. The server broadcasts all the information it has at the start of each broadcast period. Thus suppose the broadcast channel is capable of supporting $C$ bits/sec, and the server has $D_i$ bits of data to send during period $i$, for $i \geq 0$. Then broadcast period $i$ takes $R_i = D_i/C$ seconds.

From the clients' point of view, the broadcast data can be modeled as an airdisk, with each period corresponding to an airdisk rotation, where the airdisk has a data transfer rate of $C$. Since there is only one track, the airdisk has a seek time $S = 0$. If the server sends a maximum of $D_{max}$ bits during any period, the maximum rotational latency of the airdisk is $R_{max} = D_{max}/C$, since a client may have to wait up to $R_{max}$ seconds for the next rotation if it just misses the data it needed in the current rotation.

Suppose the average amount of data in any rotation is $D$. Then the average length of a rotation is $R = D/C$ seconds. Suppose also that each data item has a header, which can be read by the client to determine if the data item is of interest, and the sequence of data items on the disk is fixed, but the length of each item can vary from rotation to rotation. It might seem that the average rotational latency for a client which is reading data from the disk is $R/2$, and in fact this value has sometimes been used in previous work on periodic wireless data broadcast [14]. However, it can be shown that mean rotational latency depends not only upon the mean rotation length but also on its variance.

**Lemma 3.1** *Consider an airdisk where each item is broadcast once per rotation, the sequence of data items in every rotation is fixed, but the length of each item, and hence of each rotation, can vary. The mean rotational latency for a client accessing a given data item from such an airdisk, if the airdisk has mean rotation length $R$ and variance $s$, is*

$$\frac{R}{2} + \frac{s^2}{2R}$$

*proof.* The proof follows directly from the "residual life paradox" described in Kleinrock [12]. We sketch the underlying intuition here. Assume that a client begins reading the airdisk at at an arbitrary point of time, and is interested in item $x$. On average, item $x$ will appear every $R$ seconds. However, some of the intervals between appearances of $x$ are long, and some are short. Since the client begins reading the airdisk at an arbitrary point of time, it is statistically more likely that the client will begin reading during a long interval than during a short one. Thus the mean rotational latency experienced by the client will be more than $R/2$. (For the derivation of the formula, see [12].) Note that only if all the items were of the same length, and the rotation length were $R$ for every rotation (i.e., zero variance), would the mean latency be $R/2$. □

The mean rotational latency affects not only the access time experienced by the user, but also affects the energy consumption at the client. If the mobile client has a limited energy source (e.g. standard battery cells), energy is a very precious resource and needs to be conserved [2, 14]. In the scenario just described, suppose the clients know the sequence of data items. If, in addition, the length of the data items were fixed, the variance of the rotation length would be zero and the mean rotational latency would be reduced to $R/2$. The smaller mean rotational latency will result in reduced energy consumption at the mobile. In addition, the fixed rotation length could be used to substantially reduce the mobile's energy consumption, as follows. Initially, the client reads the preamble and period flag of one rotation. It is then able to synchronize its internal clock to that of the server, so that for subsequent reads of the airdisk, it knows where on each rotation each item appears, and waits until a very short interval before that time to start reading the airdisk. Then although the mean rotational latency remains $R/2$, the energy consumed by the mobile will be much less. In order to correct for clock drifts at the server or client, the client can maintain synchronization by periodically re-reading the preamble or period flag.

In Lemma 3.1 it was assumed that the sequence of data items on the airdisk was fixed. If the sequence of data items can change from one rotation to the next, the calculation of mean rotational latency is more complicated. First, the client has to wait until the end of the current period; this time interval equals that given in Lemma 3.1. The client then has to read the index and wait until the desired item appears on the airdisk; this second time interval depends upon the manner in which data items are scheduled for broadcast, i.e., for layout on the airdisk, at each rotation.

Note that for simplex data broadcast described in this section, the airdisk is a One-Writer, Many-Readers (OWMR) disk, where the single writer is the server. In such an airdisk the data layout on the disk may be changed by the server in response to the frequency with which the values of data items change. Thus the server may change the data layout in response to the pattern of data writes by the information provider. (This is in contrast to the situation considered in previous work [14, 5] where the data layout is changed in response to the pattern of data reads by the clients; we will consider this in the next section also.) For example, if a data item is changing rapidly the server may write it multiple times during a single rotation.

# 4 Changing data layout in response to client read patterns

We now consider the situation where the server has some information about the data items which clients are interested in. We will discuss schemes for obtaining this client interest information in sec. 5; for the moment, suppose this information is available. In that case, it may be possible to improve the performance of the airdisk system by exploiting this client interest information. For example, it is likely that some items will be more popular (i.e., of interest to more clients) than others. Imielinski et al [14] and Chiueh [5] have referred to these popular items as hot spots, by analogy with hot spots in magnetic disks, and Chiueh has essentially proposed replicating hot spots on each airdisk rotation in order to minimize the mean access time. As Chiueh has pointed out, while replication can be very useful, it consumes more wireless bandwidth and, if not performed correctly, can result in increased mean access times.

In this paper we examine another method of exploiting client interest information to improve airdisk performance, which does not rely on replication (although possibly could be used in conjunction with it.) We consider two problems of laying out the data on the airdisk. We first consider the situation in which there is no index on the airdisk (as assumed in [5]), and then consider the situation where there is an index (as assumed in [14].) Note that in the first situation client energy consumption may be high since the client has to read the airdisk all the time, while in the second situation more wireless bandwidth is consumed for the index.

**Data layout problem 1**. Suppose that the airdisk has no index but the sequence of data items can be varied at each rotation. Every item has a header identifying the item, and, for simplicity, assume that the items are all of the same length. A client then reads the airdisk, examining each item header to see if the item is of interest. In general, a client will be interested in one or more items in each rotation, and a client's access is not considered completed unless it has read all the items it is interested in. As before, suppose a client starts reading the disk at an arbitrary instant of time. The access time for a client will be the sum of four components: (1) the rotational latency, i.e., waiting for the first item the client is interested in to appear, (2) the data transfer time to read the first item, (3) the *item spread*, i.e., the time waiting until the last item the client is interested in appears, and (4) the data transfer time to read the last item. (Note that if the client is interested in more than two items, the data transfer time for those items is overlapped with the item spread and so need not be counted.) We are interested in minimizing the mean access time over all clients.

The rotational latency and data transfer times are fixed once the set of items in a given rotation is decided. However, the item spread can still be varied by varying the sequence of the items on the disk. Since client interest information is known in advance, a disk layout can be chosen to reduce the mean item spread and hence the mean access time.

We cast the problem of minimizing item spread in graph-theoretic terms as follows. Let each data item be represented by a vertex of a graph, and let the vertices be numbered consecutively. We will introduce a hyperedge for representing the items of interest to each client. Recall that a hyperedge is simply an edge that can connect more than two vertices; i.e., a hyperedge is an arbitrary subset of the set of vertices, instead of only a pair of vertices. Self-loops (i.e., hyperedges which connect only one vertex) are allowed. Thus, for each client we introduce one hyperedge which connects the vertices corresponding to the data items the client is interested in. The length of a hyperedge is the difference between its lowest and highest numbered vertex. Thus the length of a hyperedge represents the item spread for that client. The data layout problem 1 then becomes the problem of numbering the vertices so as to minimize the mean of the hyperedge lengths.

**Theorem 4.1** *The problem of minimizing the mean access time, over all clients, for data layout problem 1, is NP-complete, even if each client is interested in only two items.*

*proof.* As discussed, the mean access time is minimized by minimizing mean item spread. The problem of minimizing mean item spread can be rewritten as follows for the case where each client is interested in only two items. Given a graph $G = (V, E)$ with vertex set $V$ and edges $E$, find a one-to-one function $f : V \rightarrow \{1, 2, ..., |V|\}$ such that

$$\sum_{(u,v) \in E} |f(u) - f(v)|$$

is minimized. This problem is identical to the known NP-complete problem Optimal Linear Arrangement [1]. □

**Data layout problem 2.** Suppose the airdisk has an index which specifies the sequence of data items in each rotation, and for simplicity, assume that all the data items are of the same length. A client then accesses the data items of interest by first reading the index and then reading only each data item of interest. A client starts reading the disk at an arbitrary instant of time, and a client's access is not considered completed unless it has read all the items it is interested in. The access time for a client will be the sum of four components: (1) the rotational latency for the index, i.e., waiting for the index of the next rotation to appear, (2) the data transfer time to read the index, (3) the *item reach*, i.e., the time until the last item the client is interested in appears, and (4) the data transfer time to read the last item. (Note that as for data layout problem 1, the data transfer time for items other than the last one is overlapped with the item reach.) We are interested in minimizing the mean access time over all clients.

Once again, the problem reduces to minimizing one component, namely the item reach. We can formulate it in graph-theoretic terms using the formulation used for data layout problem 1, i.e., create a graph using the set of vertices and hyperedges as before. We can show that the item reach minimization problem is also NP-complete; however, the proof requires designing a polynomial-time reduction.

**Theorem 4.2** *The problem of minimizing the mean access time, over all clients, for data layout problem 2 is NP-complete, even if each client is interested in only two items.*

*proof.* Minimizing mean access time reduces to minimizing mean item reach. The problem of minimizing mean item reach can be rewritten as follows. Given a graph $G = (V, E)$ with vertex set $V$ and edges $E$, find a one-to-one function $f : V \rightarrow \{1, 2, ..., |V|\}$ such that

$$\sum_{(u,v) \in E} \max(f(u), f(v))$$

is minimized.

The proof is omitted here for brevity; it is given in [8]. □

Note that for the special case where every client is only interested in one item on every rotation, data layout problem 2 can be solved trivially by ordering the items in descending order of popularity, i.e., items with the most client interest first.

We are currently investigating various sub-optimal heuristics for these problems. Numerous variations and generalizations of these problems can be considered. The generalization where data items are not of the same length may make the problems significantly more complicated. It may also be possible to allow clients to specify not only their interest in a data item, but also their degree of interest. In the graphical formulation, this corresponds to assigning weights to the vertices of the graph. Such schemes require a method for making the server aware of the data items the clients are interested in. We discuss these in the next section.

# 5 Obtaining client interest information

Here we sketch various methods for obtaining information about which data items the clients are interested in. In a profile-based information service, the user service profile can be exploited. For example, consider an information service which broadcasts real-time information about road traffic information periodically throughout the day. When the user subscribes to the service a profile is set up (either by an operator or by the user via a PDA) in which the user specifies the roads, bridges etc. on the user's commute route, and the times of day that the user needs information about them (e.g., the user may be interested in knowing about traffic conditions at the Holland and Lincoln tunnels every weekday between the hours of 7:00-8:00 am and 5:00-6:00 pm, when the user commutes to and from work in New York City.) The user's commute profile is likely to be relatively static, but the user may also be able to specify special trips from time to time. In any case, the system knows sufficiently in advance the data items required by each client and when they are required, so that the server can lay out the data on the disk to try to minimize mean access time.

Another possibility is that client interest information is obtained by means of a separate mechanism where (a subset of) users are polled directly from time to time. This is how, for example, user interest in television programming is estimated, and in principle could be extended to information services also. Finally, the information service provider may use some domain-specific knowledge to estimate client information (e.g., in the road traffic example, the provider might guess that broadcasting Holland and Lincoln tunnel information as consecutive data items will help reduce mean item spread.)

# 6 RAID and AirRAID

In this section we discuss ways to increase airdisk storage capacity. The discussion here is preliminary as this work is currently in progress.

We envision that, just as for magnetic disks, new applications will place ever-increasing demands on the storage capacity of airdisks. The storage capacity of an airdisk is directly proportional to its rotational latency; if more data is to be broadcast periodically in a frequency channel of fixed size, more time will be required. Consider a system in which transmissions to clients are divided into time slots. For example, in Bellcore's WACS proposal [4] for a PCS system, transmissions to clients are divided into time slots, with eight time slots per 2.5 msec frame; each time slot is nominally capable of carrying user data at the rate of 32 kbps. Thus a time slot could also be used for sending periodic wireless broadcast data at a nominal rate of 32 kbps. Thus sending 1920 Kbits of data will require one minute;

if this is done periodically the rotational latency of the resulting airdisk is 1 minute. If the amount of data to be broadcast doubles, so will the rotational latency.

One method of increasing airdisk storage capacity is to increase the number of tracks to more than one, just as for a magnetic disk. There are potentially two ways of doing so: by increasing use of the frequency domain or of the time domain. For the time-slotted PCS system described above, in principle, the server can send the broadcast data using multiple time slots; in general, these slots need not be contiguous. Alternatively, it is possible in principle that the server can use two time slots at two different frequencies for sending periodic broadcast data. In either case, the airdisk in effect now has multiple tracks, with a corresponding increase in capacity.. Each track can be written independently by the server, and read independently by the client. The airdisk will now have a non-zero seek time penalty, in addition to the rotational latency associated with each track. Note that the practical details involved in introducing multiple tracks deserve further study; for instance, in the case where the tracks are at different frequencies, the client receiver hardware must be capable of operating at different frequencies easily.

Another possible method for increasing airdisk capacity is to borrow the idea of data striping, as is carried out in RAID for magnetic disks. Magnetic disk arrays have been proposed since the 1980s as a way of using multiple magnetic disks operating in parallel to improve aggregate I/O performance of computer systems [13, 11]. Disk array systems, commonly known as Redundant Arrays of Inexpensive Disks (RAID), are now offered in the product lines of most major computer manufacturers, and have enjoyed tremendous commercial success and applicability in high-performance workstations and supercomputers. One of the motivations for the introduction of RAID systems is to increase the storage capacity of magnetic disks in response to increased storage demands from new applications. Replacing a single magnetic disk by a single larger magnetic disk, however, is less effective in terms of performance, cost, size and power requirements than replacing it with a number of small magnetic disks with equal aggregate storage capacity.

Consider an airdisk with multiple tracks, as described above. Instead of writing data serially on each track, the server can stripe the data By analogy with the magnetic disk RAID systems, we regard each track as a separate single-track airdisk, and call the aggregate striped storage an *airRAID*. If the individual airdisks are appropriately spaced in time, it may be possible that the same client receiver hardware will be capable of reading the multiple airdisks and reassembling the bits serially. If the airdisks are on different frequencies, some receiver circuitry may need to be duplicated in each client. In either case, striping may introduce additional write latency for the server.

In our airdisk model, there are no acknowledgements of broadcast data, so forward error correction (FEC) is applied to allow the clients to recover from errors introduced by the radio channel. In the airRAID, each individual airdisk applies FEC to its data. Just as for RAID systems, multiple airdisks are less reliable than single airdisks if no FEC is applied. However, FEC on each airdisk means that, if read errors (i.e., radio channel errors) on the different airdisks are independent, the aggregate bit error rate of the airRAID will be better than for an individual airdisk. This is because a burst error on a single airdisk will not cause consecutive bit errors, making it easier for the client to recover the original data using FEC.

Note that although the discussion above has been in terms of storage capacity, use of striped airdisks which consist of the same time slot at different frequencies can potentially improve performance by allowing parallel data access, in the same manner as for magnetic RAID systems. There are several theoretical and practical issues related to airRAID, such as the type of "parity" (error correction) used and its placement, which we are currently considering in our ongoing work.

# 7 Summary

We have presented a model, called the airdisk model, for representing periodic wireless data broadcast in terms of a magnetic disk. We have shown expressions for its mean and maximum rotational latency under certain assumptions. We have then defined two problems of laying out the data on the airdisk based upon information about which items are of interest to clients. We have shown that both problems, if the data is to be laid out so as to minimize mean client access time, are NP-complete. We have considered ways of obtaining information about the items of interest to clients. We have exploited the airdisk model to suggest ways of increasing storage capacity on the air, using multiple-track disks or via ideas similar to RAID for magnetic disks.

This paper describes work which is in progress. Several issues have been related to airdisks and airRAID which deserve further study and which we are currently pursuing [8].

# References

[1] D. Adolphson and T. C. Hu. Optimal linear ordering. *SIMA J. Appl. Math.*, 25:403–423, 1973.

[2] B. R. Badrinath and T. Imielinski. Data management issues in mobile computing. In *Wireless Datacomm '92*, 1992.

[3] Bellcore. Feature description and functional analysis of Personal Communications Services (PCS) capabilities. Special Report SR-INS-002301, Bellcore, Apr. 1992.

[4] Bellcore. Generic criteria for Version 0.1 Wireless Access Communications Systems (WACS). Technical Advisory TA-NWT-001313, Issue 1, Bellcore, July 1992.

[5] T. Chiueh. Scheduling for broadcast-based file systems. In *Proc. MOBIDATA Workshop.* Rutgers University, Nov. 1994.

[6] Juan Miguel del Rosario, R. Bordawekar, and Alok Chaudhary. Improved parallel I/O via a two-phase run-time access strategy. In *Proc. Workshop on I/O in Parallel Computer Systems*, pages 56–70, 1993. Also in ACM SIGARCH Comp. Arch. News., Dec. 1993.

[7] G. Herman et al. The Datacycle architecture for very large high throughput database systems. In *Proc. SIGMOD*, pages 97–103, 1987.

[8] Ravi Jain et al. Airdisks and AirRAID. 1995. In preparation.

[9] T. F. Bowen et al. The Datacycle architecture. *Comm. ACM*, pages 71–81, Dec. 1992.

[10] M. Garey and D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. Freeman, 1979.

[11] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, San Mateo, CA, 1990.

[12] Leonard Kleinrock. *Queuing Systems Volume I: Theory*. Wiley, 1975.

[13] David Patterson, Garth Gibson, and Randy Katz. A case for redundant arrays of inexpensive disks (RAID). In *ACM SIGMOD Conference*, pages 109–116, June 1988.

[14] S. Viswanathan T. Imielinski and B.R. Badrinath. Energy efficient indexing on air. In *Proc. SIGMOD*, pages 25–36, 1994.

[15] A. Tanenbaum. *Computer Networks*. Prentice-Hall, 1988.