A Replication Schema to Support Weak Connectivity in Mobile Information Systems*

Evaggelia Pitoura

Computer Science Department, University of Ioannina, 45110 Ioannina, Greece.
Email: pitoura@cs.uoi.gr. URL: http://zeus.cs.uoi.gr/˜ pitoura/

Abstract. We present a replication schema, appropriate for distributed environments where connectivity is weak, partial or variant such as in mobile information systems. The schema is based on augmenting the database interface with operations with weaker consistency guarantees. An implementation of the schema based on distinguishing copies into quasi and core is presented and protocols for enforcing it are introduced. Then, practical examples of its applicability are described. The performance of the weak consistency schema is evaluated for various networking conditions by using an analytical model developed for this purpose.

1 Introduction

Advances in telecommunications and in the development of portable computers have provided for wireless communications that permit users to actively participate in distributed computing even while moving. The resulting distributed environment is subject to restrictions imposed by the nature of the wireless medium [5, 6, 11]. Mobile computing is susceptible to frequent network disconnections. Mobile clients deliberately avoid use of the wireless links to reduce cost and power consumption and to overcome availability and latency barriers. Thus, most mobile users are only *occasionally* connected, switching between connected and disconnected modes of operation. Similar considerations are applicable to portable computing. Other predictable disconnections include those caused by battery limitations and handoffs. Another characteristic of mobile computing is weak connectivity. Wireless networks deliver much lower bandwidth than wired networks and have higher error rates [5]. Finally, mobile systems are characterized by high variation in network bandwidth, that can shift one to four orders of magnitude, depending on whether the host is plugged in or using wireless access and on the type of its current connection [5, 6].

Mobile users will desire access to private or corporate databases that will be stored at mobile as well as static hosts and queried and updated over the wired and the wireless network. These databases, for reasons of performance, and availability will be distributed and replicated over many sites. In this paper, we propose a replication schema appropriate for environments with partial, weak

[1] To appear in the 7th International Conference on Database and Expert Systems Applications (DEXA), September 1996

and variant connectivity. Instead of requiring mutual consistency of all copies of a data item we allow bounded inconsistencies. Specifically, all data located at strongly connected hosts are grouped together to form a cluster. While all data inside a cluster are consistent, degrees of inconsistency are defined for replicas at different clusters. The proposed mechanism enchances the interface offered by database systems with operations with weaker consistency guarantees which allow access to data that exhibit bounded inconsistency. It is the responsibility of the individual application to decide between weak and strict operations so that bandwidth is utilized. Furthermore, users can still operate even when disconnected by accessing local data. Finally, the degree of inconsistency can be tuned based on the networking conditions.

The weak consistency replication schema was first introduced in [12], where its theoretical formulation was presented. In this paper, we focus on practical aspects of the model. First, in Section 2, we refine and clarify the model and present an implementation based on distinguishing copies into core and quasi. Then, in Section 3, protocols are introduced for realizing the proposed schema along with some practical examples of its use. In Section 4, an analytical model is outlined and used to study the effect of weakening consistency on the system performance. The performance parameters considered are the system throughput, the number of messages, and the response time. The study is performed for a range of networking conditions, that is for different values of bandwidth and for varying disconnection intervals. In Section 5, we present concluding remarks and a brief comparison of our work with related research.

2 A Weak Consistency Model

We consider a distributed database with data located both at mobile and stationary hosts to allow autonomous operation during disconnections. Transactions are distributed and are initiated both at mobile and at stationary hosts.

Consistency Units. As usually, a database *state* is defined as a mapping of every data to a value of its domain [2]. Data are related by a number of restrictions called *integrity constraints* that express relationships of values of data that a database state must satisfy. A state is consistent if the integrity constraints are satisfied. In contrast to traditional distributed databases where sites are normally connected, in mobile environments sites are only intermittently connected. Thus, instead of requiring maintenance of all integrity constraints we define units of consistency, called *clusters*, by partitioning the items of a database into a set of clusters Cl_i based on their location, so that data in strongly connected sites belong to the same cluster. Then, we provide a weaker form of consistency as follows.

Definition 1. A cluster state is consistent iff all intracluster integrity constraints hold. A database state is *d-consistent* iff all cluster states are consistent and all intercluster integrity constraints are *d-degree consistent.*

The definition of *d-degree consistency* for an integrity constraint depends on its type. In this paper, we focus on replication constraints, where copies x_i of the same data item x are expected to have the same value. The d-degree relation bounds the divergence among the values of the copies of a data item and can be expressed in terms of the time lag or fuzziness of value of the copies in deviating from mutual consistency [1]. The values of the data copies are occasionally reconciled to obtain a mutual consistent value. Reconciliation is performed for instance periodically, at times of higher network availability, or when required for keeping the divergence inside the d-bound. The degree may vary based on the availability of network bandwidth by allowing little deviation in instances of higher bandwidth availability and high deviation in instances of low bandwidth availability thus making applications able to adjust to the limitations of the communication environment. The cluster configuration is dynamic. By taking advantage of the predictable nature of disconnections, clusters may be explicitly created or merged upon a forthcoming disconnection or connection of the associated mobile host. To accommodate migrating locality, a mobile host may move to a different cluster upon entering a new cell.

Weak and Strict Transactions. To maximize local processing and reduce network usage, we allow direct access to locally, e.g., in a cluster, available d-consistent data by introducing *weak reads* and *weak writes*. These operations allow users to operate on d-consistent data when the lack of strict consistency can be tolerated by the application semantics. We call the standard read and write operations strict.

Definition 2. A transaction is a partial order (OP, $<$), where OP is the set of weak and strict read, weak and strict write, abort and commit operations executed by the transaction, and $<$ represents their execution order. The partial order must specify the order of conflicting data operations and contains exactly one abort or commit operation which is the last in the order. Two weak (strict) data operations *conflict* if they access the same copy of a data item and at least one of them is a weak (strict) write operation. A *weak* transaction (WT) is a transaction where OP does not include strict operations and a *strict* transaction (ST) one where OP does not include weak operations.

Weak transactions access data copies that belong to the same cluster. User transactions are decomposed into a number of weak and strict subtransactions units according to the degree of consistency required by the application. There are two commit events associated with each weak transaction, a *local commit* in its associated cluster and an implicit *global commit* at reconciliation. Local commitment is expressed by an explicit commit operation. Updates made by locally committed weak transactions are visible only by weak transactions in the same cluster. These updates become permanent and visible by strict transactions only after reconciliation when local transactions become globally committed.

Implementation. To implement this schema we distinguish the copies of a data item into core and quasi. *Core* copies are copies whose values are up-to-date and permanent, while *quasi* copies are copies whose values may be obsolete and are only conditionally committed. To process the operations of a transaction, the database management system translates operations on data items into operations on copies of those data items. We formalize this procedure by a *translation function h*. Function h maps each $SR[x]$ operation into a number of read operations on core copies of x and each $WW[x]$ operation into a number of write operations of local quasi copies of x. Depending on the translation of a weak read operation we define two types of translation functions: a *best-effort* translation function maps each $WR(x)$ operation into a number of read operations on the most up-to-date locally available core or quasi copies of x, and a *conservative* translation function maps each weak read into a number of read operations only on locally available quasi copies. In addition, based on the time of propagation of updates of core copies to quasi, we define two types of translation functions: an *eventual* translation function maps a $SW[x]$ into writes of only core copies, while an *immediate* translation function also updates the quasi copies of the corresponding cluster. For an immediate h, conservative and best-effort have the same result. How many and which core or quasi copies are actually read or written when a database operation is issued on a data item depends on the coherency algorithm used. Without loss of generality, we assume that there is only one quasi copy per cluster. Since all quasi copies in a cluster have the same value, this single copy can be considered to be their representative.

Correctness. A complete intracluster schedule, IAS, is an observation of an interleaved execution of transactions in a given cluster configuration that includes locally committed weak transactions and (globally) committed strict transactions (for a formal definition, see [12]). Given a schedule S, the *projection of S on strict transactions* is the schedule obtained from S by deleting all weak operations, and the *projection of S on a cluster Cl_k* is the schedule obtained from S by deleting all operations of S that do not access Cl_k. A correct concurrent execution of weak and strict transactions must maintain d-consistency among clusters and strict consistency inside each cluster. The following theorem provides a serializability-based criterion of correctness.

Theorem 3. *Given that d-degree is maintained, an intracluster schedule S is correct if its projection on strict transactions is equivalent to a serial one-copy (1C) schedule S_{1C}, and each of its projections on a cluster Cl_i is conflict-equivalent to a serial schedule S_{S_i} such that the order of transactions in S_{S_i} is consistent with the order of transactions in S_{1C}.*

Protocols for maintaining the d-degree relation are given in Section 3. Any coherency control method that guarantees one-copy serializability can be used for synchronizing core copies. The schema reduces to one-copy serializability when only strict transactions are employed.

3 Weak Consistency in Practice

Serializability Protocols. We distinguish between protocols for coherency and protocols for concurrency control. Coherency control refers to ensuring that all copies of a data item have the same value, here we must maintain this property globally for core copies. Concurrency control refers to maintaining consistency of the other integrity constraints, here the intracluster constraints. For coherency control, we adopt a quorum-based schema [2]. Each strict transaction reads q_r core copies and writes q_w core copies per strict read and write operation where $q_r + q_w > n_d$, and n_d is the number of available core copies of the data item. To maintain correctness we employ *strict two phase locking* where each transaction releases its locks upon commitment [2]. Weak transactions release their locks upon local commitment and strict transactions upon global commitment. There are four lock modes (WR, WW, SR, SW) corresponding to the four data operations. Lock compatibility is defined based on the conflict relation, where two lock modes are compatible if the corresponding operations do not conflict. The lock compatibility for an eventual and best-effort h is depicted in Table 3.

	WR	WW	SR	SW
WR	X		X	
WW			X	X
SR	X	X	X	
SW		X		

Table 1. A X entry indicates that the lock modes are compatible.

Protocols for Bounding Inconsistency. The degree for each data item at a cluster expresses the divergence of the local quasi copy from the value of the core copy that may result either from globally uncommitted weak writes or from updates of core copies that have not yet been reported at the cluster. Thus, the degree may be bounded either by limiting the number of weak writes pending commitment or by controlling the h function, as outlined in Table 3 for various definitions of the divergence bound d.

Examples. Consider the case of users working on a common project using mobile hosts. Groups are formed that consist of users who work on similar topics of the project. Clusters correspond to data used by people in the same group who need to maintain consistency among their interactions. We consider data that are most frequently accessed by a group as data *belonging* to this group. At each group, copies of data items belonging to the group are core copies, while copies of data items belonging to other groups are quasi. A data item may belong to more than one group if more than one group frequently accesses it. In this case, core copies of that data item exist in all such clusters. In each cluster, operations on items that do not belong to the group are weak, while operations on data that belong to the group are strict. Weak updates on a data item are accepted only when they do not conflict with updates by the owners of that data item. Clustering can also be used to model caching in a client/server architecture. In such a setting, a mobile host acts as a client interacting with a server at a fixed

Definition of d	Protocol
Up to d transactions can operate on inconsistent data.	The number of weak transactions at each cluster is bounded appropriately.
d is the maximum number of divergent versions per item	The function h is defined such as a strict write modifies quasi copies at a cluster at least after every d updates. This definition cannot be ensured for involuntary disconnected clusters since there is no way of notifying them remote updates of core copies.
(for data items with arithmetic values) Each copy can take a value only inside a range d of acceptable values	Only weak writes whose values are in the allowable bounds are accepted.
Up to d data items are allowed to diverge.	The number of quasi copies that are allowed to diverge is bounded to d by allowing weak writes only on a set of d data items.
Up to d data copies per item are allowed to diverge.	The number of quasi copies that are allowed to diverge at each cluster is bounded so that the total number of quasi copies that differ from core copies in all clusters is d. This is achieved by bounding appropriately the number of weak writes at each cluster.

Table 2. Possible definitions for bounded inconsistency.

host. Data are cached at the client for performance and availability. Cached data are quasi copies, while data at the fixed host are core copies. Transactions initiated by the server are always strict. Transactions initiated by the client that invoke updates are always weak while read-only client transactions can be strict if strict consistency is required. At reconciliation, weak writes are accepted only if they do not conflict with strict transactions at the server. The frequency of reconciliation depends on the user consistency requirements and on networking conditions. As a final example, consider data representing the location of a mobile user. Such data are fast-changing. They are also frequently accessed to locate a mobile host, thus must be replicated at many sites to reduce the overhead of searching. Most of the location copies should be quasi. Only a few core copies are always updated to reflect changes in location.

4 Performance Evaluation

To evaluate the proposed schema and quantify the improvement in performance attained by sacrificing strict consistency in weakly connected environments, we have developed an analytical model. The analysis follows an iteration-based methodology for coupling standard hardware resource and data contention as in [14]. Data contention is the result of concurrency and coherency control. Resources include the network and the processing units. We generalize previous results to take into account (a) nonuniform access of data, that takes into consideration hotspots and changing locality, (b) two different transaction types, weak and strict, and (c) various types of data access, as indicated by the compatibility matrix of Table 3. An innovative feature of the analysis is the employment of a vacation system to model disconnections of the wireless medium.

We assume a cluster configuration with n clusters and a Poisson arrival rate for both queries and updates. Let λ_q and λ_u be respectively the average arrival rate of queries and updates on data items initiated at each cluster. Let c be the *consistency factor* of the application under consideration, that is c is the fraction of the arrived operations that are strict. To model hotspots, we divide data at each cluster into hot and cold data sets. To capture *locality* we assume that a fraction o of the transactions exhibit locality, that is they access data from the hot set with probability h and data from the cold set with probability $1-h$. The remaining transactions access hot and cold data uniformly. Due to mobility o may diminish with time. Locality is taken into consideration by the replication schema, by assuming that the probability that a hot data has a core copy at a cluster is l, and that a cold data has a core copy is l', where normally, $l' < l$. For a detailed description of the model see [10].

We study the effect of the consistency factor on the system performance. The performance parameters under consideration are the system throughput, the number of messages sent, and the response time of weak and strict transactions. The study is performed for a range of networking conditions, that is for different values of bandwidth and for varying disconnection intervals. Table 3 shows some realistic values for some of the input parameters. For the bandwidth the values depend on the type of technology used, for infrared a typical value is 1 Mbps, for packet radio 2 Mbps, and for cellular phone 9-14 Kbps [5]. We assume an eventual and best effort h.

Parameter	Description	Value
n	number of clusters	5
λ_q	query arrival rate	12 queries/sec
λ_u	update arrival rate	3 updates/sec
t_u	processing time for an update	0.02 sec
t_b	propagation overhead	0.00007 sec
V	vacation interval	ranges
W	available bandwidth	ranges
m	average size of a message	16 bits
D_c	number of cold data items per cluster	800
D_h	number of cold data items per cluster	200
N	average number of operations per transaction	10

Table 3. Model parameters.

System Throughput. Figures 1(left), 1(right), and 2 show how the maximum transaction input, or system throughput, is bounded by the processing time, the available bandwidth, and the disconnection intervals respectively. We assume that queries are four times more common than updates $\lambda_q = 4\,\lambda_u$. As shown in Figure 1(left), the allowable input rate when all transactions are weak ($c = 0$) is almost double the rate when all transactions are strict ($c = 1$). This is the result of the increase in the workload with c caused by the fact that strict operations on data items may be translated into more than one operation on data copies. The percentage of weak transactions can be effectively tuned to attain the desired throughput based on the networking conditions such as the duration of disconnections and the available bandwidth. As indicated in Figure 1(right), to get for instance, the same throughput with 100bps as with 500bps and $c = 1$ we

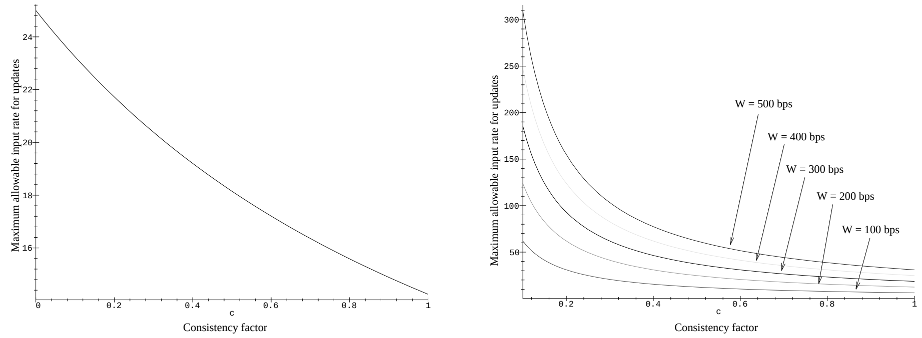

Fig. 1. Maximum allowable input rate for updates. (left) Limits imposed by the processing rate at each cluster. (right) Limits imposed by bandwidth restrictions.

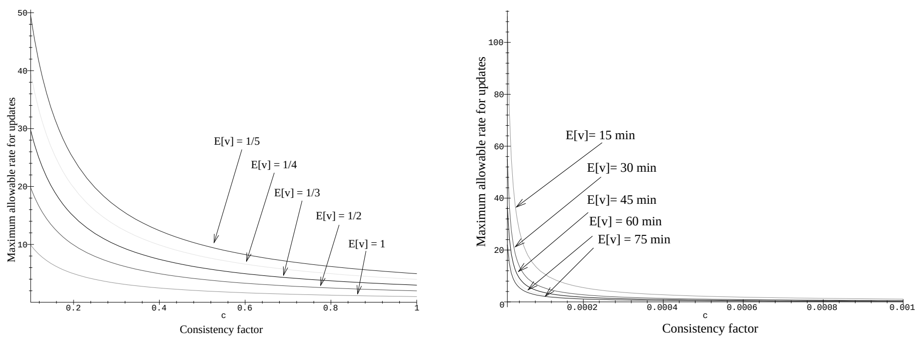

Fig. 2. Maximum allowable input rate for updates. Limits imposed by disconnections.

must lower the consistency factor below 0.1. The duration of disconnections may vary from seconds when they are caused by handoffs to minutes for instance when they are voluntary. Figure 2 depicts the effect of the duration of a disconnection on the system throughput for both short durations (Figure 2(left)) and longer ones (Figure 2(right)). For long disconnections (Figure 2(right)), only a very small percentage of strict transactions can be processed. To keep the throughput comparable to that for shorter disconnections (Figure 2(left)) the consistency factor must drop at around three orders of magnitude.

Communication Cost. We estimate the communication cost by the number of messages sent. We assume a ROWA schema ($q_r = 1$, $q_w = n_d$) if not otherwise stated. As shown in Figure 3(left) the number of messages increases linearly with the consistency factor. As expected the number of messages decreases with the percentage of transactions that access hot data, since then local copies are more frequently available (Figure 3(middle)). Thus the communication overhead in mobile computing is increased due to the frequently changing locality. The number of messages decreases when the replication factor of hot core copies increases (Figure 3(right)). The decrease is more dramatic since most operations are queries and the coherency schema is ROWA, thus for most operations no messages are sent. On the contrary, the number of messages increases with the replication factor of cold core copies because of additional writes caused by coherency control (Figure 4(left)). Finally, since most operations are queries keeping the read quorum smaller than the write quorum decreases the number of messages (Figure 4(right)).

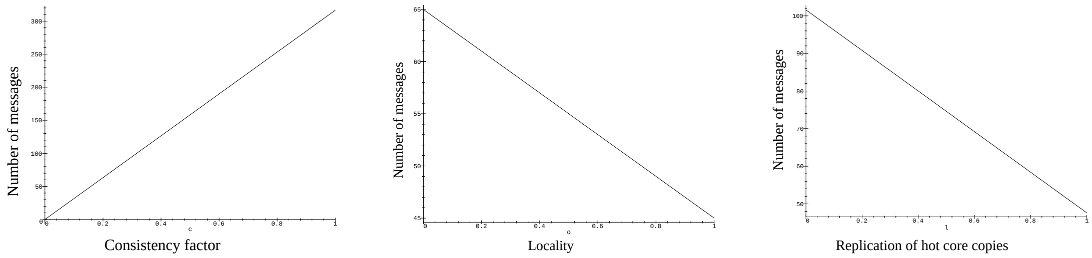

Fig. 3. (left) Number of messages for various values of c. (middle) Number of messages with locality (right) Number of messages for different replication of cold core copies.

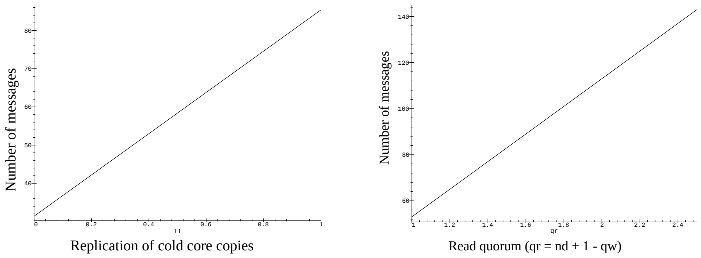

Fig. 4. (left) Number of messages for different replication of hot core copies. (right) Number of messages for different values of the read quorum.

Transaction Response Time. The response time for weak and strict transactions for various values of c is depicted in Figure 5. The larger values of response times are for 200bps bandwidth, while faster response times are the result of higher network availability set at 2Mbps. The disconnection intervals are exponentially distributed with mean $E[V] = 1/5$ sec, to model disconnection intervals that correspond to short involuntary disconnections such as those caused by handoffs. The coherency control schema is ROWA. The latency of weak transactions is about 50 times greater than that of strict transactions. However, there is a trade-off involved in using weak transactions, since weak updates may be aborted later. The time to propagate updates during reconciliation is not counted. As c increases the response time for both weak and strict transactions increase since more conflicts occur.

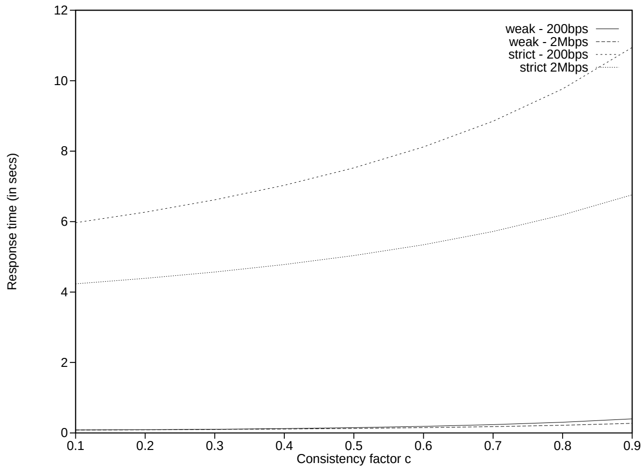

Fig. 5. Comparison of the response times for weak and strict transactions.

5 Conclusions and Related Work

To overcome bandwidth, cost, and latency barriers, clients of mobile information systems switch between connected and disconnected modes of operation. In this paper, we propose a replication schema appropriate for such operation. The schema is based on extending the database interface with weak operations. Mobile clients can operate even when disconnected using weak operations. Bandwidth can by utilized by deliberately using weak transactions. In addition, the degree of consistency can be appropriately tuned to achieve the desired system performance. Weak operations offer a form of *application-aware adaptation* [9]. Application-aware adaptation characterizes the design space between two extremes. At one extreme, adaptivity is entirely the responsibility of the application, that is there is no system support or any standard way of providing adaptivity. At the other extreme, adaptivity is subsumed by the system, here the database management system. Weak and strict operations lie in an intermediate point between these two extremes, serving as middleware between a database system and an application.

The partitioning of a database into clusters resembles the *network partition problem* [4], where site or link failures fragment a network of database sites into isolated subnetworks called partitions. Clustering is conceptually different than partitioning in that it is electively done to increase performance. Furthermore, whereas all partitions are isolated, clusters may be partly connected. In the following, we compare briefly our work with the most closely related research. For an extensive comparison see [10]. The *Bayou system* [13] is a platform of replicated mobile databases on which to build collaborative applications. A read-any/write-any weakly-consistent replication schema is employed. Each Bayou database has one distinguished server, the primary, which is responsible for committing writes. The other secondary servers tentatively accept writes and propagate them towards the primary. Each server maintains two views of the database: a copy that only reflects committed data and another copy that also reflects tentative writes currently known to the server. Applications may choose between committed and tentative data. Tentative data are similar to our quasi data, and committed data similar to core data. Correctness is defined in terms of groups of read and write operations, called sessions, rather than on serializability. The *Coda* file system [7] treats disconnections as network partitions and follows an optimistic strategy. No degrees of consistency are defined and no transaction support is provided. [8] extends Coda with a new transaction service called *isolation-only transactions* (IOT). IOTs are sequences of file accesses that unlike traditional transactions have only the isolation property. IOTs are similar to weak transactions. In [3], where a weak read operation is added to a *file service interface*. The semantics of operations are different in that no weak write is provided and since there is no transaction support, the correctness criterion is not based on one-copy serializability.

References

1. R. Alonso, D. Barbara, and H. Garcia-Molina. Data Caching Issues in an Information Retrieval System. *ACM TODS*, 15(3):359–384, 1990.
2. P. A. Bernstein, V. Hadjilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addisson-Wesley, 1987.
3. D. Tait C and D. Duchamp. Service Interface and Replica Management Algorithm for Mobile File System Clients. In *Proceedings of the 1st International Conference on Parallel and Distributed Information Systems*, pp. 190–197, 1991.
4. S. B. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in Partitioned Networks. *ACM Computing Surveys*, 17(3):341–370, 1985.
5. G. H. Forman and J. Zahorjan. The Challenges of Mobile Computing. *IEEE Computer*, 27(6), 1994.
6. T. Imielinksi and B. R. Badrinath. Wireless Mobile Computing: Challenges in Data Management. *Communications of the ACM*, 37(10), 1994.
7. J. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM TOCS*, 10(1):3–25, 1992.
8. Q. Lu and M. Satyanarayanan. Improving Data Consistency in Mobile Computing Using Isolation-Only Transactions. In *Proceedings of the 5th Workshop on Hot Topics in Operating Systems*, 1995.
9. B. D. Noble, M. Price, and M. Satyanarayanan. A Programming Interface for Application-Aware Adaptation in Mobile Computing. In *Proceedings of the 2nd USENIX Symposium on Mobile and Location-Independent Computing*, 1995.
10. E. Pitoura. *Transaction Management in Mobile Heterogeneous Environments*. Ph.D. Thesis, Department of Computer Science, Purdue University, August 1995.
11. E. Pitoura and B. Bhargava. Building Information Systems for Mobile Environments. In *Proceedings of the 3rd International Conference on Information and Knowledge Management*, pp. 371–378, 1994.
12. E. Pitoura and B. Bhargava. Maintaining Consistency of Data in Mobile Distributed Environments. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, pp. 404–413, 1995.

13. D. Terry, A. Demers, K. Petersen, M. Spreitzer, M. Theimer, and B. Welch. Session Guarantees for Weakly Consistent Replicated Data. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, pp. 140–149, 1994.
14. P. S Yu, D. M. Dias, and S. S Lavenberg. On the Analytical Modeling of Database Concurrency Control. *Journal of the ACM*, 40(4):831–872, 1993.