ISSN: 2321-7782 (Online)
Volume 3, Issue 4, April 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Mobile Computing and Databases

Sneha J. Tripathi[1]
Computer Science and Engineering
H.V.P.M COLLEGE of Engineering and Technology
Amravati, India

P. L. Ramteke[2]
Information Technology
H.V.P.M COLLEGE of Engineering and Technology
Amravati, India

Abstract: ***The emergence of powerful portable computers, along with advances in wireless communication technologies, has made mobile computing a reality. Among the applications that are finding their way to the market of mobile computing - those that involve data management hold a prominent position. In the past few years, there has been a tremendous surge of research in the area of data management in mobile computing. This research has produced interesting results in areas such as data dissemination over limited bandwidth channels, location-dependent querying of data, and advanced interfaces for mobile computers. This paper is an effort to survey these techniques and to classify this research in a few broad areas.***

Keywords: Mobile computing, databases, data dissemination, bandwith, location-dependent queries, interfaces, transaction management.

I. INTRODUCTION

MOBILE computing has become a reality thanks to the convergence of two technologies: the appearance of powerful portable computers and the development of fast reliable networks. In the mobile wireless computing environment of the future [33] massive number of low powered computer machines will query databases over the wireless communication channels. Mobile clients will often be disconnected for prolonged periods of time due to the battery power saving measures; they will also frequently relocate between different cells and connect to different data servers at different times. The mobile or nomadic computing environment no longer requires users to maintain a fixed and universally known position in the network and enables unrestricted mobility of the users. Mobility and portability will create an entire new class of applications and possibly new massive markets combining personal computing and consumer electronics. In this paper, we survey the impact that mobile computing has had in the area of data management. We first analyze each of the distinctive features of mobile computing and how they impact the implementation of databases for mobile computers, creating new opportunities for research. Then for each of these problems we survey the papers that have appeared in the literature in recent years. A cell could be a real cell as in cellular communication network or a wireless local area network which operates within the area of a building. In the former case the bandwidth will be severely limited (supporting data rates on the order of 10 to 20 Kbits/sec.). In the latter, the bandwidth is much wider up to 10 Mb/sec. Fixed hosts will communicate over the fixed network, while mobile units will communicate with other hosts (mobile or fixed) via a wireless channel. Although a wireless network with mobile clients is essentially a distributed system, there are some characteristic features that make the system unique and a fertile area of research. These are:

Asymmetry in the communications: The bandwidth in the downstream direction (servers-to-clients) is much greater than that in the upstream direction (clients-to-servers). Moreover, in some systems, the clients do not have the capacity to send messages to the servers at all. As we mentioned above, even the bandwidth in the downstream direction is limited: 10 to 20 Kbits/sec. in a cellular network and up to 10 Mb/sec. in a wireless LAN.

***Frequent disconnections*:** Mobile clients do not stay connected to the network continuously (as fixed hosts do), but rather users switch their units on and off regularly. Moreover, mobile clients can roam, disconnecting from a cell to connect to another.

***Power limitations*:** Some of the portable units are severely limited by the amount of energy they can use before the batteries have to be recharged.

***Screen size*:** Some of the portable units, such as the Personal Digital Assistants, have very small screens. Each one of these features has an impact on how data can be effectively managed in a system with mobile clients. The communication asymmetry, along with the restriction in power that the mobile units have, make the model of broadcasting data to the clients (instead of waiting for the clients to request specific data items), an attractive proposition. This is called data dissemination and is the topic of Section 2 in this paper. The limited bandwidth and the pattern of frequent disconnections have a clear impact on how transaction management is implemented and how data consistency is guaranteed in this environment. This is the subject of Section 3. The roaming of clients through different cells opens the possibility of answering queries in a way that is dependent on the current position of the client, i.e., implementing location dependent queries. We review the work in this area in Section 4. Finally, screen and power limitations have an impact on the kind of interfaces that can be implemented for data browsing and querying. We cover the work in this field in Section 5. Section 6 offers some conclusions about the future of this research area.

II. DATA DISSEMINATION

Data dissemination is the delivery of data from a set of producers to a larger set of clients. This is characterized by an inherent asymmetry in the communications: The bandwidth in the downstream direction (servers-to-clients) is much greater than in the upstream direction (clients-toservers). (For a survey of data dissemination work This model fits very well a mobile system, since mobile clients are usually unable to transmit data at a very high speed (sometimes this capacity is not even present). However, they can receive data at high rate. The model of sending information to a client population, without waiting for specific requests is also called push-based. By pushing data, the servers avoid interruptions caused by requests and are allowed to propagate data whose existence would otherwise be ignored by clients. On the other hand, push-based systems have the problem of deciding about the relevance of the data to be broadcasted to clients. The usefulness of the system depends on the ability to predict the clients needs. A simple solution is to allow clients to subscribe to services while providing profiles of their interest [49]. The clients subscribe to information providing queries that describe their interest. These queries can be viewed as continuous queries. The server also needs to decide whether to send the data periodically or aperiodically. Periodic push has the advantage of allowing clients to disconnect for certain periods and still not miss out items (since they will be broadcasted again after the client is reconnected). Aperiodic dissemination on the other hand is a more effective way of using the bandwidth available. Periodic push has been used in the past in many systems, Acharya et al. have proposed the use of a periodic dissemination architecture in the context of mobile systems. They call the architecture *Broadcast Disks*. Broadcast Disks are novel in two ways:

- » They provide a multilevel mechanism that permits data items to be broadcast nonuniformly, so more bandwidth can be allocated to items according to their relative importance.
- » Mechanisms for managing the storage in the clients are devised to tailor caching and prefetching to perform efficiently with the multilevel broadcast.

Using Broadcast Disks, one can construct a memory hierarchy in which the highest level contains a few items and broadcasts them with high frequency while subsequent levels contain more and more items and broadcast them with less and less frequency. In this way, one can establish a trade-off between access time for high-priority data and that of the low-priority items.

Fig. I. An example of broadcast disks.

Fig. I shows an example in which two levels are provided. One can think of this broadcast as being provided by two disks: the first containing data item *A* and the second items *B*1, *B*2, and *B*3. The first disk rotates at a speed three times as fast as the second. The characterization of broadcast programs was presented. A good program renders approximately fixed interarrival times for each item and allocates bandwidth to items according to their access probabilities. An effort in optimizing programs is described. Since no broadcast program can perfectly match the needs of an individual client, the need to compensate for the mismatches exists. This is done by doing intelligent caching and prefetching at the client site. Caching and prefetching policies for Broadcast Disks have been studied in. The mechanisms have been extended to incorporate the effect of updates. In this last case, the server must effectively communicate updates to the client by means of invalidation or propagation. Acharya et al. studied the way of devising prefetching algorithms that work well under update scenarios. Acharya et al. extend their previous work to study the integration of a pull-based and a push-based Broadcast Disk approach. In a pull-based operation, clients explicitly request items by sending messages to the server, which in turn sends the information back to the clients. Combining the two approaches can be beneficial for both server and clients. Their study assumes two independent data channels, front and backchannel: the front channel is used for push-based operation, while the back-channel serves as the media for the pull-based operation. The available bandwidth is shared between these two channels. The study concludes that for the extreme cases of little contention at the server or server overload, pure-pull and pure-push solutions respectively are the best options. For the other cases, the authors propose a hybrid solution, called Interleaved Push and Pull (IPP), which allows clients to use the backchannel to send pull requests for items that do not appear in the Broadcast channel, while the server supports Broadcast Disks plus interleaved responses to the pulls in the frontchannel. IPP suffers of the same bottleneck problems of any pull approach. To improve the scalability of IPP, the authors propose three different techniques:

1. Adjusting the pull bandwidth, at the expense of the push bandwidth. This introduces a trade-off between how fast the queue of pull requests is served versus how fast the broadcast delivery is executed. Lower values (30 percent of the available bandwidth) for the pull bandwidth do not produce the best results for lightly loaded systems, but they also do not produce the worse results for saturated systems, and thus they are good candidates for dynamic loads.

2. Providing a pull threshold. With this technique, clients only request their most expensive misses, i.e., those that will be broadcast in the longest future. The effect of reducing the number of pulls coming from clients is to delay the point at which the server queue saturates.

3. Chopping off the slowest part of the broadcast schedule successively larger pieces. This has the effect of increasing the available bandwidth for pull.

This introduces the disadvantage that if there is still not enough bandwidth for pull, performance can degrade considerably since clients will not be able to pull the items that were not broadcast. The authors discuss an algorithm to rapidly adapt the content of mobile client's caches based on the misses that result in on-demand data requests. The algorithm is developed as a way to effectively support data dissemination using Direct Broadcast Satellite systems. Their simulation results show that the heuristic of adapting caches based solely on information about the miss ratio works quite reasonably.

The IRs are very succinct ways of transmitting this information. For instance, one could send a list of identifiers for the items that have changed since the last Invalidation Report was sent. Depending on the technique used, IRs can cause "false negatives," making a client drop an item from its cache when in reality the item is still valid. In return for this anomalous behavior, one gets a shorter report that consumes less bandwidth. The authors study the tradeoffs involved in using different types of Invalidation Reports. The issue of relaxing consistency of the caches is addressed. For instance if the mobile clients are

caching stock prices, it may be perfectly acceptable to use values that are not completely up to date, as long as they are within 0.5 percent of the true prices. This can be accomplished by considering the cached values as *quasicopies* of the values in the server [6]. A quasicopy is a cached value that is allowed to deviate from the true copy in a controlled way. Using quasicopies, the IRs can be made even shorter, thereby saving extra bandwidth for their transmission. Also, the authors extend the IRs strategies to dynamically adjust to changes in the query and update rates and the disconnection patterns of the mobile clients. These new techniques, called adaptive IRs, take into account the on demand queries that the clients make when they do not find the items in their local caches. In Wu et al. the problem of discarding caches in mobile computers that have been disconnected for any period of time (sleepers) is addressed. The strategy a mobile unit that disconnects is to purge its cache after it comes back in operation. Wu et al. propose a mechanism to decide whether some items in the cache can still be used by the mobile unit. To do so, the mobile units need to contact the server when they come back from a disconnection and ask if the items in the cache are still usable. To decrease uplink traffic, the database is partitioned in groups and elements in the same group are cached together. In this way, the mobile unit only asks the server for the validity of specific groups. The server keeps track of the update history for elements in each group, and using an algorithm called Grouping with Cold Update-Set Retention (GCoRe), it determines whether or not all the items updated since the mobile computer became disconnected have been included in the last IR. If the answer is yes, the cached group can be kept (since the mobile unit can hear about the updates in the IR). Otherwise, the cached group should be discarded (there is not enough information in the IR to determine which elements to keep). Using simulation, the authors determine that as the group size increases, the uplink traffic decreases but the downlink traffic increases. The total bandwidth usage first decreases and then increases suggesting an optimal operating point. Liu and Maguire use IRs to propose a mobilityaware dynamic database caching scheme for mobile computing. The IR techniques are combined with knowledge of the mobile user's behavior to broadcast IRs only within the user's mobility area. Using a performance model, the authors claim the scheme can reduce the system cost by more than 87 percent when compared to a fully replicated database system. The previous Broadcast Disks protocols are not designed with real-time demands on mind. In real-time environments, minimizing the average latency time is no longer the main criteria for performance. Rather, guaranteeing that time constraints are met is the important concern. Also, in the previous Broadcast Disks protocols, failures are ignored. The basic premise for that is that errors in fetching data as it becomes available in the channel are deemed as tolerable, since the data will become available when it is rebroadcast. For real time systems, however, this assumption may imply missing critical deadlines. To remedy these two problems, Bestavros proposes a series of Broadcast Disk organizations that are suitable for real-time environments and the usage of an Adaptive InformationDispersal Algorithm (AIDA), which adds reliability to the broadcast. First, Bestavros presents three different organizations: flat, rate monotonic and slotted rate monotonic which are suitable for real-time environments. In flat organizations, the simplest one of them, the server sends the union of the data items needed by its client in a periodic fashion. With a flat broadcast, there is the need of guaranteeing that the worst-case latency will be tolerable even for those items that have the strictest deadlines. In order to guarantee that, enough bandwidth should be allocated. This organization, is therefore, wasteful of bandwidth. To remedy this, the rate monotonic organization broadcasts each item at a rate inversely proportional to its time constraint. So, each item represents an independent disk "spinning" at a rate reciprocal to the timing constraint associated with the item. This is obviously an extreme organization that tries to optimize bandwidth at the expense of ease of broadcast programming. The slotted rate monotonic organization achieves a balance between the previous two organizations by coalescing data items together so they could be put into the same broadcast disk. This coalescence is achieved by partitioning the data set into bins. Items in the same bin share the same broadcast disk and therefore, have identical worst-case latencies. Fault-tolerance is achieved by Bestavros using AIDA, an elaboration on the Information Dispersal Algorithm (IDA) of Rabin [53]. Information Dispersal takes a file F and divides it in N independent pieces (dispersal) in such a way that recombining *any* m of such pieces ($n \geq N$) is sufficient to retrieve F (reconstruction). AIDA uses IDA to adaptively decide the amount of redundancy used in the broadcast. Namely, the number of pieces transmitted n is allowed to vary from m (no redundancy) to N (maximum redundancy). Fig. 3, taken illustrates how two files A and B are broadcast using

IDA. File *A* is dispersed into 10 blocks, of which any five blocks are enough to reconstruct it. For file *B* the algorithm uses *N* = 6, *m* = 3. The pieces are broadcast periodically, using a program as shown in Fig. 4. There are two "periods" in that program: the broadcast period, which extends for eight units and, in the absence of errors is enough to recover both files; and the program data cycle which broadcasts all the pieces from all the dispersed files in the disk. Bestravos finds upper bounds for worstcase delays given the number of transmission errors. If the data is just broadcast without the any form of directory, clients have to tune to the channel continuously until the required data items are downloaded. This forces clients to be in active mode for a long time, consuming a lot of power, which is already scarce. By providing a directory, the server enables the ability of tuning and that in turn allows clients to become active only when the data of interest is being broadcast. Clients can remain in *dozemode* most of the time, thereby saving energy. The authors state the concept of *Indexing on Air*, i.e., transmitting an index along with the data, so clients can tune only during the times they need to. One issue that arises is how the index is multiplexed with the data to make the latency and tuning time minimal. Sending an index with every data broadcast increases the latency since clients that miss the index have to wait for the next broadcast. Broadcasting the index *m* times during each data broadcast one can minimize the probability of missing the index. This is called (1, *m*) indexing. The optimal value of *m* along with the analysis of the technique is provided. This technique can be further optimized by realizing that there is no need to replicate the entire index between data segments. It is enough to send the portion of the index relevant to the data segment that follows it. This is called *Distributed Indexing* . Distributed indexing results in approximately the best latency and tuning time. It is also possible to have secondary indexes on air, which are described. All these techniques, however, do not take in consideration the access frequency of different data items, or data skew in the construction of the index. Reducing the average number of index probes can have a significant impact in power consumption, so using the skew to guide the structure of the index is a good idea. This is the motivation of Chen et al. where imbalanced indexes that favor most frequently accessed items are studied. The indexing techniques can be used to disseminate data using *temporal addresses*; that is, the client is given the exact time when the relevant data item is going to be published. Temporal matching is performed by the clock circuitry in the client, consuming little energy. However, as a drawback,this method requires a high degree of clock synchronization, forcing the network to be deterministic. This assumption is reasonable for exclusively used channels such as Motorola's Satellite Networks (Embarc) where only one server controls the channel. In general, however this is not easily achieved (consider for instance and Ethernet where the disseminating traffic can be mixed up with on-demand traffic). For these cases, the techniques have to be slightly modified. One solution is to have clients tuning up e slots ahead of time [60]. Alternatively, data dissemination can be achieved by using *multicast addresses*. The server sends data to a group of clients using the same address. In IP, for instance, clients can use 32 bit addresses starting with 1111. Equipped with an Ethernet card, a client can match the predefined set of addresses and keep the CPU in doze mode until the data's multicast address matches the address specified by the client. Clients join multicast groups and filter the relevant data using the Ethernet card. Hashing can be efficiently used in combination with multicast addresses. The client computes the hash value of the data item of interest and sets the interface card to listen to all broadcast values on the multicast address corresponding to the hash value just computed. The hash function can be given to the client upon registration. A randomized algorithm that minimizes the expected latency is described. If the number of items exceeds the number of multicast addresses, the tuning time for accessing an item depends on the distribution of items among addresses. An algorithm that optimally partitions the data is provided.

Broadcast schedules to minimize latency have been studied. The results obtained there can be combined with schemes to reduce tuning time as well. The main contributions of that study are a rule that characterizes the minimal latency and three algorithms to schedule broadcasts. The rule shows that the latency is minimized when the frequency of an item is proportional to the square-root of its demand probability and inversely proportional to the square-root of its size. The first scheduling algorithm is an on-line algorithm that selects the item for which the expression $(T - R(i))2pi/li$ is the largest. In the expression, T is the current time, $R(i)$ is the last time at which item i was transmitted most recently, pi is the demand probability for item i and li the size of item i. This decision criteria can be computationally expensive. To decrease the computational demands, another

on-line algorithm is proposed In this algorithm, items are divided in buckets which are kept as cyclical queues. The algorithm chooses to broadcast the first item in the bucket for which the expression $(T\,R(Im))2qm/dm$ is the largest. In the expression *Im* is the item at the front of the bucket *m*, while *qm* and *dm* are the average demand probability and average size for the items in bucket *m*. Simulations show that the buckets algorithm performs close to optimal, specially as the number of buckets is increased. Notice that the notion of buckets can be compared to the notion of Broadcast Disks The main differences, however are the facts that the buckets algorithm is on-line and the algorithms insists in placing the instances of items equally spaced in the disk. Finally an off-line algorithm is introduced by using an on-line algorithm to generate an a priori schedule of an specified size *N*.

Some recent applications of data dissemination include information dissemination in the Internet, and in private networks, Advanced Traveler Information Systems , and dissemination using satellite networks.

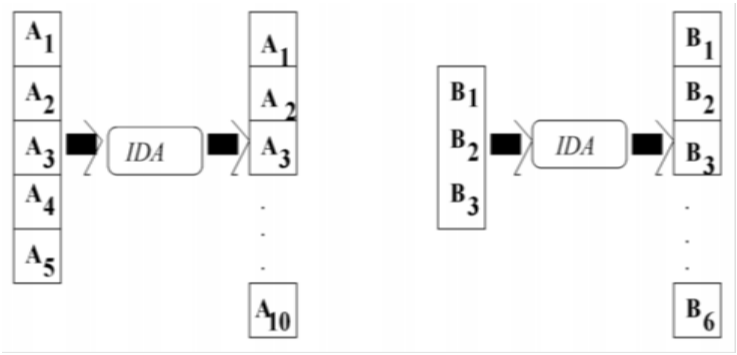

Fig. II. Dispersal of two files

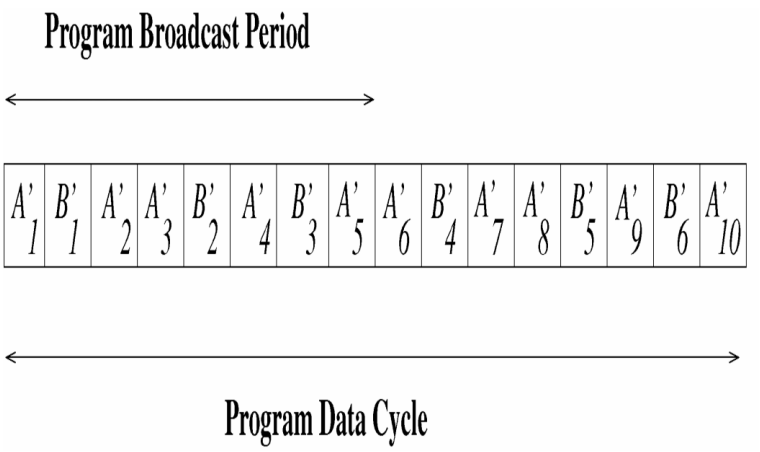

Fig. III. Broadcast program

III. DATA CONSISTENCY

Consistency guarantees for data processed by mobile clients are an important area of research. These guarantees provide the basis for any collaborative work and transaction processing done with these systems. The authors present a mechanism to provide individual applications with a view of the database that is consistent with their own actions. This is important since in their environment, clients can read or write data from any one of the available servers and these servers can contain inconsistent views of the database. For example, a mobile client of a distributed database could issue a write at one server and later on read the item at a different server, obtaining an inconsistent value if the two servers have not been synchronized between the two operations. The paper introduces *session* guarantees that alleviate this problem. A session is defined as a sequence of read and write operations performed during the execution of an application by a client. The guarantees provided by their method are:

» *Read Your Writes*: Any read operation in the session must reflect the values established by previous writes in that session.

» *Monotonic Reads*: Successive reads reflect a nondecreasing set of writes.

» *Writes Follow Reads*: Writes are propagated after the reads on which they depend.

» *Monotonic Writes*: Writes are propagated after writes that logically preceded them.

These guarantees are enforced by having a session manager that serializes read and write operations and by assigning unique identifiers to each write. A server must be willing to return information about the unique write identifier and the set of write identifiers for writes that are relevant to a given read. The session manager maintains a readset of identifiers for the writes relevant to session reads and a write-set of identifiers for those writes performed in the session. For instance, the Read Your Writes guarantee is enforced by following two steps. Whenever a write is accepted by a server, its identifier is added to the session's write-set. Before a read is processed in a server *S* at a time *t*, the session manager must check that the write-set is a subset of the sequence of writes received by *S* before *t*. These session guarantees were developed to be deployed in the prototype implementation of the Bayou project, which aims to provide a collaboration framework between mobile computing users. Krishnakumar and Jain discuss a method to perform mobile sales transactions using site-transaction escrow methods. Escrow methods divide the total number of available instance of an item among the number of sites in the system. A transaction can successfully complete at a site if the number of instances it requires does not exceed the number available at that site. By using a reconfiguration protocol, such as the demarcation protocol , sites can negotiate an increase of the allotted fragment. A mobile client starts performing a transaction in one server, using escrow methods. If the client needs to move to another server during the life of the transaction (because it moved from one geographic area to another), the only information needed at the new server is the operation to be performed next. (Since the old server made its decisions based on escrowed resources.) At the end of transaction, however, a two-phase commit algorithm is needed to synchronize the two servers. This protocol is therefore very suitable for clients that do a fair amount of mobility during the life of the transaction. Walborn and Chrysanthis generalize the usage of escrow techniques by exploiting object semantics to facilitate autonomous and disconnected operations in mobile database applications. The basic idea is to split large and complex objects into smaller fragments and cache a set of these fragments on a mobile host. By making these fragments the unit of consistency and defining consistency constraints to be enforced in the fragments, one can have concurrent operations on a set of mobile computing clients. Using the demarcation protocol , one can ensure that local executions are serializable. The protocol does this by exploiting the object organization and constraints. When a mobile client requires access to an object, it sends a request to the database server invoking a *split* operation that selects part of the object and establishes some consistency conditions. This part of the object is then only accessible by transactions on the mobile client that requested it. The rest of the object remains available for other clients. Examples of objects that can be subjected to fragmentation are: aggregate items (e.g., plane seats), sets and stacks. For instance, sets can be split into a number of subsets and these subsets recombined in an arbitrary order to reconstruct the original set. Each mobile host can specify a range of elements as the selection criteria, with the ranges being disjoint. Gray et al. propose a two-tier replication algorithm that allows mobile applications to propose tentative transactions at the mobile client. These tentative transactions are performed over the data replicas stored at the clients while these clients are disconnected. When the mobile client reconnects, the tentative updates are reprocessed as base transactions in the node that stores the master copy of the data. The tentative transactions may fail when reprocessed (due to conflicts with other transactions). If that happens, the mobile client (and user) that originated the transaction is informed about the failure and why it happened. Processing the transaction in the master node may render different from the tentative results. This may be acceptable for some applications (checking account balances may change because other transactions may have affected the balance) and unacceptable for others (an item is out of stock and cannot be sold). This technique does a good job in solving a fundamental issue: standard ways of propagating updates to replicas (eager replication, lazy replication make deadlocks increase as the cube of the number of sites and as the fourth power of transaction size, thereby rendering the systems unscalable. On the other hand, the two-tier technique may result in an unacceptable number of failed transactions (after the clients reconnect) and user dissatisfaction. The concept of *Certification*

Reports is introduced as a way of supporting transactions in a mobile environment. The technique uses the broadcast channel to help mobile clients do some of the work of verifying if the transactions being run by them need to be aborted. Clients do that bylistening to Certification Reports that contain lists of items that are in the read and write set of active transactions that have declared their intention to commit to the server during the previous broadcast period. If the client sees a conflict between the reports read and writesets and its own read and writesets, it aborts the transaction (this is inspired by optimistic concurrency control algorithms . Otherwise, when the transaction is ready to commit, the client sends to the server the read and writesets along with the values to be written in the database. The server does the final verification to check whether two transactions running on different clients collide with each other. The server then acknowledges accepted and rejected transactions using the broadcast channel. One of the crucial advantages of this protocol is that it downloads most of the work of validating transactions to the clients. Moreover, transactions which ultimately would be rejected get an early abortion at the client site and can be resubmitted promptly (as opposed to the situation created by the technique. Lu and Satyanarayanan propose a new transaction model called *isolation only transactions* (IOT), with which the system can provide strong consistency while not guaranteeing other traditional transaction properties, such as atomicity and durability. The execution model uses optimistic concurrency control. As in the previous techniques, a transaction is executed in the mobile client and no partial execution is visible on the servers. When the transaction completes, it enters a *committed* or *pending* state. If the execution of the transaction does not contain any partitioned data access (i.e., the client maintains server connection for every item that has accessed), the transaction is committed and its results made visible on the servers. Otherwise, the transaction enters the pending state and waits to be validated later. If the validation succeeds, its results are reintegrated and committed in the servers. Otherwise, the transaction is resolved (manually or automatically). In order to do the validation, the technique applies Davidson's optimistic transaction model . This method builds a precedence graph to represent interdependencies among transactions and detects cycles in the graph that are indicative of transactions that do not satisfy global serializability. The last issue we address in this section is that of data replication. The ability to replicate data in mobile environments is essential since disconnected units can continue to process objects that are stored locally. The authors point out that even though the number of copies in a system can be large, the key issue is how to manages copies that are *updateable*. This copies are also called the *core* copies and their number is likely to be small for a simple reason: it becomes too expensive to update a large number of core copies. The number of read-only copies (or cached copies) can be large, but these copies may not be current. The core set can be managed used standard quorum algorithms but very often the core set needs to be reconfigured. For instance, a person may be editing a document first from her home computer, then from a laptop on a train ride and then from the office computer. At each step a copy of the document is transferred from one computer to the next. At every step the copy in the computer being used currently is the core set. Wang and Pâris describe a protocol to facilitate the reconfiguration of core sets. The protocol relies in the introduction of nodes that act as *referees*. These nodes are responsible for the updating and storage of the core set description. Referee nodes do not participate in data accesses and do not hold replicas of the objects. Their solely mission is to record changes in the composition of the core set. Once a replica-holding node decides that it needs to change the core set, it sends the description of the new set to the referees and if a majority of them accept the change, that description becomes the new core. Referees can be highly replicated since they are only accessed when a new core is elected. As it is pointed out in, the directory of objects is also a replicated object that has to be managed just like any other. The authors conclude that standard techniques to manage copies are also adequate to manage the directory. However, a variation of the primary copy algorithm, called *Primary by Row* is proposed to manage directories. In a Primary by Row algorithm, each core copy has the ability to modify the portion of the directory that concerns the copy. For instance, in the example above, the core copy of the traveling document should be able to modify the row of the directory that points to the current location of the document. A number of researchers are currently working in providing replication in weakly connected systems, mainly using the Distributed Shared Memory paradigm ([45]). Among them, the work in [59] focuses in providing support for very low-level objects of arbitrary sizes. The authors of [38] describe how to use *Lazy Release Consistency* [39] (which is based on the fact that real consistency is not needed until a synchronization

situation is reached), to develop a programming model for large distributed systems with computing nodes that can be weakly connected.

IV. LOCATION DEPENDENT QUERYING

The fact that clients in a mobile environment can change locations enables the possibility of answering queries in a way that is dependent on the current position of the client. This is the subject of this section. Imielinski and Badrinath [33] presented the concept of queries with location constraints, i.e., constraints which involve location of the mobile computing users. For instance the query "find me a doctor near my house" has a unary constraint on location. The query "find X, Y, and Z such they are in the same building and Y is between X and Z" involves a ternary constraint (between) plus three unary constraints involving individual locations (in the same building). The main problem becomes how to minimize the communication cost to retrieve the necessary information to answer the query. Naive strategies result in too many messages and long latencies. On the other hand, building optimal plans can be shown to be NP-complete. The authors suggest using greedy heuristics based on the ID3 [32] algorithm to solve the problem. Imielinski and Navas discuss a family of protocols that integrate Global Positioning System (GPS)1 into IP to enable the creation of location dependent services [34]. Examples of these services are: multicasting messages selectively to specific geographical locations (such as a train station), providing services to clients within a certain geographical,range from the server and providing information for mobile users when that information depends on the user's location. Genesis and Advanced Traveler Information Systems (ATIS) [54], [55] are systems being developed to facilitate travel. Travelers are mobile users equipped with personal communication devices and portable computers. ATIS provides updated information on weather and road conditions, detours, construction zones, bus schedules, and parking. This information is available prior and during the trip. The users can as a variety of queries, such as recommendations of the most favorable routes from a starting location to a chosen destination. The term favorable may represent minimum travel time, shortest distance, etc. Genesis is an ATIS operational test project developed in Minnesota as a partnership between the University of Minnesota, the Minnesota Department of Transportation, and private sector companies. The Genesis system includes: data collection stations such as the Traffic Management Center (TMC) and Metropolitan Transportation Commission (MTC), the fixedend ATIS database server which stores the data provided by the TMC and MTC along with the history of travel times, bus schedules and fares and other static information, a wireless communication service provider and mobile personal communication devices. Data is either disseminated by the server as a result of a predetermined transaction or sent at the request of a mobile client. User queries include requests for travel time, roads under construction in a given area, shortest path for a trip, route evaluation and emergency service requests. Events, such as traffic accidents and road hazards are monitored and trigger data dissemination to travelers that need to be warned. Typical triggers warn travelers about congestion and alternative routing. Mobisaic [65] is a World Wide Web information system designed to serve mobile users. Mobisaic extends the WWW by allowing documents to refer and react to the current location of clients. It does so by supporting two new concepts: *Dynamic Uniform Resource Locators* (*URLs*) and *Active Documents*. A URL is dynamic if it contains at least a dynamic environment variable that gets substituted with the environment context (e.g., location). For instance a user gets a hyperlink to a document that describes the user's current location in a building by placing a dynamic URL in the corresponding WWW that contains the link. When a user selects a dynamic URL in a document, the client browser is responsible for resolving all references to dynamic environment variables within the URL. After this resolution, the browser has a standard URL which can be sent to the server to get the corresponding page. Active documents are HyperText Markup Language (HTML) documents that enable the client to automatically react to changes in a user's mobile computing context. Authors write active documents just like they write standard HTML documents, with the addition of a subscribe command that lists the dynamic environment variables that the client must subscribe when it loads the document. When the variables listed in the command change value, the client must take an action that reloads the current page (with new variables) or another page (marked by a new URL). Spreitzer and Theimer [57] describe an architecture to manage location information. In this architecture, personal information is managed by User Agents, while a partially decentralized Location Query Service is used to facilitate location-based operations. There is a

User Agent for each user. The agent collects and controls all personal information regarding its user. Applications can only get personal information through a user's agent. The sources of information collection for an agent include infrared-based active badges [68], GPS, motion sensors and cameras, and human sources.

V. CHALLENGES

Although the volume of work in this area has been large during the recent future, many challenges still remain. We point to some of them in this section:

» *Prototyping*: In the area of data dissemination, many theoretical studies, simulations, and focused implementations have been tried. However, a full scale prototype that encompasses all of the main ideas is still missing.

» *Bandwidth utilization*: A comprehensive study that analyzes the best way to divide the bandwidth between all the options (data broadcast, indexes, invalidation reports, answers to queries and uplink capacity) is needed. The study in [3] is a good step towards understanding the trade-offs, but more work is needed.

» *Transactional properties*: More work is needed in studying real cases where transactional properties are needed in mobile applications. Moreover, studying which properties can be effectively relaxed (or simply ignored) for real applications, as long as effective protocols to enforce the remaining properties is an area that could have a very practical impact in the field.

» *Optimization of location dependent query processing*: Very little has been done in finding ways to obtain quick, and perhaps approximate, answers to these queries.

» *Data visualization*: A key issue is to effectively use the scarce display space in mobile computers to present answers to queries.

VI. CONCLUSION

Mobile computing has proven a fertile area of work for researchers in the areas of database and data management. The inherent limitations of mobile computing systems present a challenge to the traditional problems of database management. As we can see, the amount of research in this area in the last few years have been staggering. However, some problems remain open for research. There is a need for better protocols in the area of data sharing and transaction management, better interfaces, clever algorithms that exploit locality to shape the answers to queries. Undoubtedly, we will continue to see a steady number of research contributions in the future.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their helpful comments to improve this paper.

References

1. S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-Based Data Delivery Using Broadcast Disks," Personal Comm., vol. 2, no. 6, Dec. 1995.
2. S. Acharya, M. Franklin, and S. Zdonik, "Prefetching from A Broadcast Disk," Proc. 12th Int'l Conf. Data Eng., New Orleans, 1996. [3]_ S. Acharya, M. Franklin, and S. Zdonik, "Balancing Push and Pull for Data Broadcast," Proc. ACM SIGMOD Int'l Conf. Management of Data, Phoenix, Ariz., pp. 183-194, May 1997.
3. S. Acharya, M. Franklin, S. Zdonik, and R. Alonso, "Disks on Air," Proc. ACM SIGMOD, 1994.
4. S. Acharya, M. Franklin, S. Zdonik, and R. Alonso, "Broadcast Disks: Data Management for Asymmetric Communication Environments," Proc. ACM SIGMOD Int'l Conf. Management of Data, San Jose, Calif., May 1995.
5. R. Alonso, D. Barbará, and H. Garcia-Molina, "Data Caching Issues in An Information Retrieval System," ACM Trans. Database Systems, vol. 15, no. 3, Sept. 1990.
6. R. Alonso and V.S. Mani, "A Pen-Based Database Interface for Mobile Computers," Proc. First IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, Calif., Sept. 1994.

