# REPLICATION MANAGEMENT AND OPTIMISTIC REPLICATION CHALLENGES IN MOBILE ENVIRONMENT

[1]Archana Sharma
[1]Institute of Technology and Science, Mohan Nagar, Ghaziabad
asharma269@rediffmail.com

[2]Dr. Vineet Kansal
[2]Institute of Technology and Science, Mohan Nagar, Ghaziabad
kansalvineet@hotmail.com

## *ABSTRACT*

*Wireless and mobile communication is an important technology that improves user's daily life and facilitates the development of some new technology such as electronic commerce and mobile commerce. Replication is a fundamental technique for supporting concurrent work practices in mobile environments. Optimistic replication algorithms allow replica contents to become stale but in a controlled way. In return, they become far more efficient and available than traditional replication algorithms that keep all the replicas consistent, especially when the network and computers are unreliable. The use of optimistic replication has grown explosively due to the proliferation of the use of the Internet and mobile computing devices, but its systematic study has begun only recently. This paper explores the various replication strategies, replication in mobile environment, benefits and challenges of optimistic replication in mobile applications.*

## *Keywords*

*Replication, DBMS, DB2, FTP, SDR, usenet*

## 1. INTRODUCTION

Managing data in a mobile computing environment invariably involves caching or replication. In many cases, a mobile device has access only to data that is stored locally, and much of that data arrives via replication from other devices, PCs, and services. Given portable devices with limited resources, weak or intermittent connectivity, and security vulnerabilities, data replication serves to increase availability, reduce communication costs, foster sharing, and enhance survivability of critical information.

A replication scheme determines the number and location of replicas in a distributed system. Traditional static replication schemes do not perform well in mobile environment since the assumptions of fixed hosts and relatively static access patterns no longer hold. Data replication in wireless environment enhance network performance, availability and reliability. As wireless

DOI: 10.5121/ijdms.2011.3407 

communication systems are organized the serviced area into cells or zones, where each cell is serviced by a base station which is located at the centre of a cell. The interface to the wired network is achieved through wired links which connect the base stations to the mobile switching centres attached to the wired network. The mobile users are assigned to a centre according to their geographical location and can communicate with a base station via the wireless channels. The information regarding a user's location is replicated at each centre to improve access delay for routing a call, however this replication causes problems in updating databases.

# 2. LITERATURE REVIEW

Data replication involves copying data from one source location to another target location (Elmasri and Navathe, 2004; Gollmick, 2003; Wiesmann et al, 2000) to guarantee that transactions captured on the source database are propagated to one, or more target databases in an orderly manner (Rennhackkamp, 1997). The traditional role of data replication has been to increase availability and performance of distributed databases through the provision of fail over configurations and elimination of the need to access remote distributed data sites. Because of rapid increases in data and network expansions, organizations have reverted to storing data on different geographical distributed sites for operational efficiency (Rennhackkamp, 1997).

A fundamental question in replicated data system is how to handle updates to multiple copies of the same data item. If the copies cannot communicate instantaneously then concurrent updates to different replicas of the same data item are possible violating the ideal semantics of emulating single copy data storage. Conservative update replication systems prevent all concurrent updates causing mobile users who store replicas of data items to have their updates rejected frequently particularly if connectivity is poor or non-existent. Even when connected mobile users will spend bandwidth to check consistency at every update. Conservative strategies are often appropriate in the wired world, but they work poorly in most mobile environments.

Data replication strategies are divided into optimistic and pessimistic approaches(J.Gray, P. Helland, 1996, M. Wiesmann, 2000, P. Bernstein, 1997). Pessimistic replication avoids update conflicts by restricting updates to a single replica based on the pessimistic presumption that update conflicts are likely to occur. This ensures data consistency because only one copy of the data can be changed. Pessimistic replication performs well in local- area networks in which latencies are small and failure uncommon. Primary copy algorithms(A. Hetal, 1996) are an example of pessimistic approaches. However , pessimistic approaches are not suitable for mobile environments, because they are built for environments in which the communication is stable and hosts have well known locations.

An optimistic replication, in contrast, allows multiple replicas to be concurrently updatable based on the optimistic presumption that update conflicts are rare. Conflicting updates are detected and resolved after they occurred. Therefore, this schema allows the users to access any replica at any time, which means higher write availability to the various sites. However optimistic replication can lead to update conflicts and inconsistencies in the replicated data (Arshaf Ahamed, 2010)

Using optimistic replication in mobile environment has been studied in several research efforts. ROAM (D. Ratner, P. Reiher, 2004) is an optimistic replication system that provides a scalable replication solution for the mobile user. ROAM is based on the on the Ward Model (D. Ratner,

P. Reiher , 2001). The authors group replicas into wards ( wide area replication domains). All ward members are peers, allowing ant pair of ward members to directly synchronize and communicate.

A multi-master scheme is used in (J. Monreiro, A.Brayner, 2007) that is, read-any/write-any. The servers allow access (read and write) to the replicated data even when they are disconnected. To reach an eventual consistency in which the servers converge to an identical copy , adaptation in the primary commit scheme is used.

A hybrid replication strategy is presented in( J.Adbawajy, M. Deris 2006) that have different ways of replicating and managing data on fixed and mobile networks. In the fixed network, the data object is replicated to all sites, while in the mobile network, the data objects is replicated asynchronously at only one site based on the mostly visited site.

Cedar(N. Tolia, M. Satyanrayanan, 2007) uses a simple client- server design in which a central server holds the master copy of the database. At infrequent intervals when a client has excellent connectivity to the server( which may hours or days apart), its replica is refreshed from the master copy.

# 3. DATA REPLICATION STRATEGIES

## 3.1 Synchronous Data Replication

Under synchronous data replication (SDR) strategy, updates are applied to all the database replicas of an object as part of the original transaction. The database replicas are then kept in a state of synchronization at all the network nodes by updating all the replicas as a part of one atomic transaction . That is, when one copy is updated, all other copies need to be updated as well. . By means of locking protocols a conflict can be avoided and 1-copy serializability is guaranteed. (Wiesmann et. al 2000, 14.) Synchronous replication comes with a cost but it guarantees that all copies have the same data. In write-all approach (Bernstein et. al, 1987, 266), if a transaction tries to write to all copies of replicated system and one copy is not available, the transaction can't commit. This doesn't provide availability and prevents replication occurring. To avoid situation like this, a technique called voting is used. When voting is used, the majority of copies must be able to write and read. However, this strategy of data replication is slow and prone to deadlocks and hence organizations have to contend with stale data versions from time to time, or implement programs to reconcile these data. By mandating concurrency control, SDR eliminates integrity anomalies during replication.

## 3.2 Asynchronous Data Replication

The mechanics of synchronous replication might prevent execution of an update, leaving rows locked for a period of time. Asynchronous replication might be the preferred solution. (Ramakrishnan & Gehrke 2002, 621.) With asynchronous replication in a multi-master of peer-to-peer environment, an update is done in one database, and when it is committed the transaction is propagated to other masters. This way it requires less networking and hardware resources than synchronous replica-tion. However, ASDR does not enforce consistency between the replicated databases.

A conflict might occur, as for a period of time, data might be different in different master sites. In Oracle for example, for transaction a remote procedure call is created by a trigger and it's placed on deferred transaction queue. On a scheduled or on-demand interval, the de-ferred transactions are propagated to destinations. At the destination server the transaction is applied and checked for conflict. If conflict is detected, Oracle has resolution methods availa-ble. (Oracle 2009.)

### 3.3 Snapshot Replication

Snapshot replication is in all replication types as initial base copy of the database to subscribers. A snapshot is a copy of a database in a single point of time. Entire snapshot is initially copied from the distributor to subscribers by the distribution agent. From thereon, the data in the published tables is refreshed periodically. This can be a long process as database snapshots can be large in size and can take lot of resources. Therefore careful planning and scheduling is in order. Generally, snapshot replication is used mostly in read-only data distribution setting, where data is updated infrequently. It is the simplest replication type SQL Server offers. (Thomas & McLean 2006, 531.)

### 3.4 Push and Pull Data Replication Strategies

Push data replication strategy includes both snapshot replication and near-real time replication schemes. Snapshot data replication is best suited to applications not in need of current data (Wiesmann and Schiper, 2005). Such applications are found in data warehousing, or data mining as well as non-real time decision support systems. A near real time replication employs triggers that are stored at each local database and executes each time a part of the replicated database is updated propagating the changes to the other remote databases. In the push strategy, the source data controls the data replication procedures while with pull data replication schemes, the local databases determine the replication processes. These schemes are preferred because they are less disruptive.

## 4. MOBILE DATABASE

Mobile database provides access to a large amount of data through mobile communication. Mobile database capture data and access data wherever you are. Instantly collect, retrieve and review critical data regardless of physical location Imielinski *et al*. (1997) stated that mobile users would be in constant need of stock information, traffic directions, local directory, weather information etc. Wireless medium would be used as the first mile of the information highway that would disseminate massive amounts of information across the country. Organizing and accessing data on wireless communication channels are new challenges to the database and telecommunication communities deployment to companies would help to:

- streamline and improves work flow and business process
- Increase productivity
- Automate collections, new business, customer service and financial transactions
- Reduced costs
- Improve customer service
- Provides on spot decisions support whenever you need it.

In mobile computing environment users can access information through wireless connections regardless of their physical location. In mobile database systems, new features such as mobility, disconnection, low bandwidth, high bandwidth variability, heterogeneous networks and security risks make traditional database processing schemes no longer well suited.

## 4.1 Architecture of Mobile Transaction Environment

An important feature of these database systems is their ability to allow optimistic replication of data by permitting disconnected mobile devices to perform local updates on replicated data. Replication is one of the key technologies in promoting the performance of mobile database systems. Replication is a general technique to increase the data availability. One key feature of these database systems is their ability to deal with disconnection. Disconnection refers to the condition when a mobile system is unable to communicate with some or all of its peers. In such a situation, the mobile no longer has access to shared data. To deal with the disconnection problem, optimistic replication approaches have become exceedingly common.

In such approaches, the mobile unit is allowed to locally replicate shared data and to operate on this data while it is disconnected. The local updates can be propagated to the rest of the system on reconnection. However, since the local updates potentially conflict with other updates in the system, schemes to detect and resolve such conflicts are required.

Traditionally, distributed database is located in a fixed decentralized network. A mobile database can be defined as the union of distributed database, disconnected database ad-hoc database and broadcast disks (Xia and Helal, 2003). The distributed database is treated as the home of mobile database and the others deal with the access of mobile users (Fig. 1). According to Xia and Helal (2003), Traditional database design is static and limits the flexibility of database applications while mobility is changing the way we design databases and their DBMS.

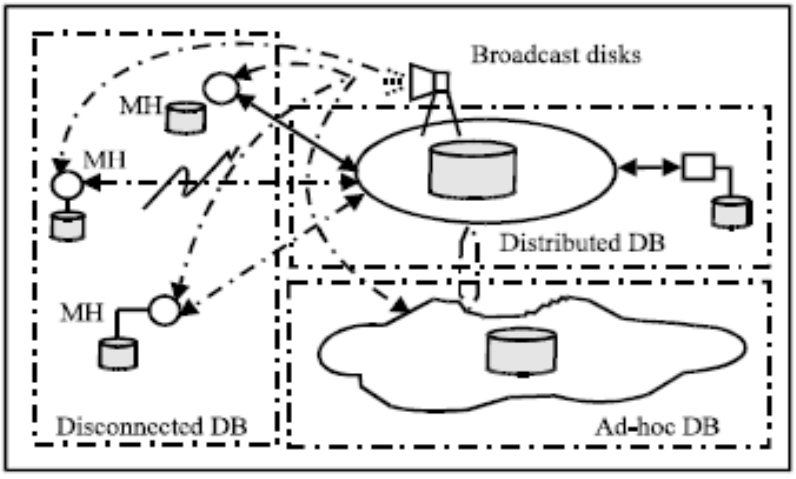


Figure 1: Architecture of Mobile Transaction Environment

In mobile database, everything is dynamic, varying from sporadic accesses by individual users to particular data to continuous access of a particular data by a large group of user. This is the case from disconnected database access to broadcast disks. Mobile hosts have to deal with planned or

unexpected disconnections when they mobile; they are likely to have scarce resources such as low battery life, slow processor speed and limited memory; their applications are required to react to frequent changes in the environment such as new location, high variability of network bandwidth; their data interests are changing from time to time and from location to location; even data semantics in mobile hosts are varying according to data access patterns, connection duration and disconnection frequencies etc.

# 5. COMPARATIVE STUDY : USAGE OF DATA BASE SOFTWARE IN MOBILE TRANSACTION

The following products are surveyed: Microsoft SQL Server CE, Oracle Lite, and IBM DB2 Everyplace and focused on describing in detail how these commercial products support mobile transactions and how data consistency is achieved.

## 5.1 Microsoft SQL CE

The Microsoft SQL Server CE (SSCE) [Mic] is a client-agent-server architecture that supports database applications on mobile hosts (see Figure 2). The database on a mobile host is a small replicated portion of the main database. When mobile hosts are disconnected, transactions are processed locally at the mobile hosts. When mobile hosts reconnect to the database server, synchronization processes are carried out to reconcile information. The client agent at the mobile host connects to the server agent through the Internet Information Server (IIS) that resides on the database server. This means that the role of mobile support stations is not an issue in SSCE systems.

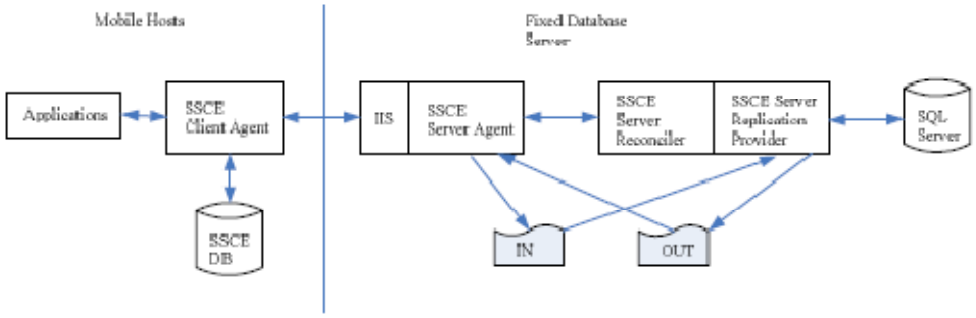


Figure 2: Microsoft SQL CE architecture

**Transaction properties:** The Microsoft SQL Server CE supports both flat and nested transactions at mobile hosts. Sub-transactions only reveal committed results to the parent transaction. When the top-level transaction commits, the results are visible to local transactions at the mobile host. Transactions at mobile hosts are executed sequentially.

**Data consistency*:*** When the mobile host reconnects to the database server, a synchronization process is performed to reconcile information. The client agent sends all changes in the local database to the server agent. The server agent, then, writes the updates to a new input file and initiates a reconciliation process at the SQL Server Reconciler. The reconciliation process will detect and resolve conflicts. Different conflict resolutions are supported in the SSCE system, for example priority based or user defined. When the reconciliation process completes, it will inform

the SQL Server Replication Provider to finally write the successful updates to the database server. When there are updates at the database server, an inverse process is carried out to propagate these updates to the mobile host.

## 5.2 Oracle Lite

Oracle Lite [Ora] is a client-server architecture that makes use of a replicated copy of the main database (which is called a snapshot) to support disconnected transaction processing at mobile hosts (see Figure 3). Oracle Lite does not include mobile support stations in its architecture. The replicated database system at the mobile host is called a snapshot that can be read-only or updatable. When the mobile host is disconnected from the database server, transactions are processed locally. The snapshot is synchronized with the master copy at the database server when the mobile host reconnects.

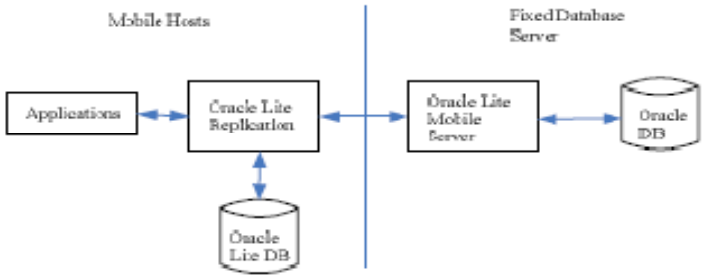


Figure 3: Oracle Lite architecture

**Transaction properties:** The Oracle Lite only supports flat transactions at mobile hosts.

**Data consistency*:*** When the mobile host connects to the database server, a refresh process will be performed to synchronize the snapshot with the master copy. If the snapshot is modified, the updates will be sent to the database server. All local transactions at the mobile host will be validated at the database server in the same order as they were executed at the mobile host. The refresh process is a blocking process. This means that no database operations will be allowed at the mobile host during the reconciliation process.

## 5.3 IBM DB2 Everyplace

IBM DB2 Everyplace [IBM] is an architecture that consists of a relational database at mobile hosts and a mid-tier on fixed hosts. The mid-tier system supports data synchronization between the mobile databases that reside at the mobile hosts with the source databases on the fixed database servers. When mobile hosts are disconnected, transactions are processed locally at the mobile hosts. When mobile hosts reconnect to the database server, synchronization processes are carried out to reconcile data. As Microsoft SQL Server CE and Oracle Lite, IBM DB2 Everyplace does not discuss mobile support stations. Data synchronization processes are carried out directly between the mobile hosts and the fixed database servers.

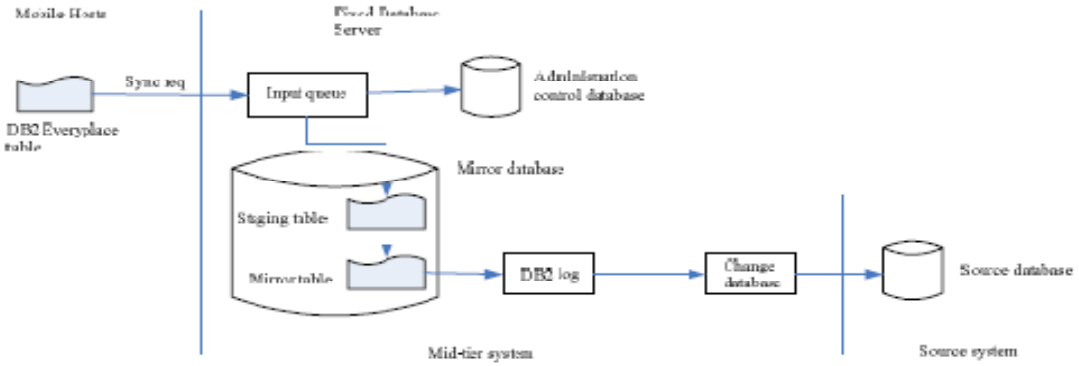


Figure 4: IBM DB2 Synchronize from mobile hosts to fixed hosts

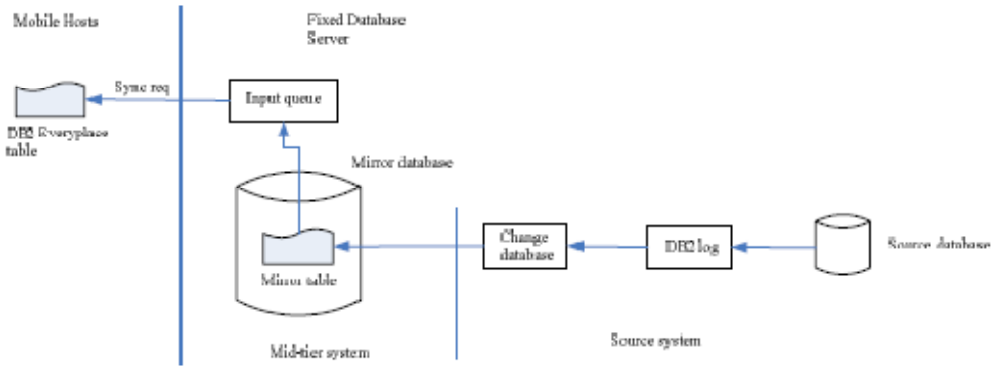


Figure 5: IBM DB2 Synchronize from fixed hosts to mobile hosts

**Transaction properties:** The IBM DB2 Everyplace only supports flat transactions.

**Data consistency*:*** When the mobile host connects to the database server, a synchronization process will be performed to synchronize data between the mobile hosts and the source database. IBM DB2 Everyplace differentiates the data synchronization processes between the mobile host and the source database. The data synchronization from the mobile host to the source database is illustrated in Figure 4. The synchronization request is submitted from the mobile host and placed in the input queue at the fixed database server. If the synchronization request is allowed to proceed, the data at the mobile host is temporarily saved in the Staging table then the Mirror table. If there is any conflict, it will be resolved in the Mirror table. After this, the changes are stored in the DB2 log and sent to the source database through a Change Data table. For the data synchronization from the source database to the mobile host, an inverse process is performed (as illustrated in Figure 5). The main difference between these two data synchronization processes is that the data from the source database is immediately processed and transferred to the mobile host without any delay, i.e., without passing through the Staging table and Administration control.

# 6. REPLICATION IN MOBILE ENVIRONMENT

A replication system for mobile use must provide four basic qualities:

1. Support for any-to-any communication
2. Support for potentially hundreds of read-write replicas
3. A file- granularity replication-control interface
4. A “get-up and go" model requiring no pre-motion actions.

By definition, mobile users change their geographic location. Since it is typically cheaper, faster, and more efficient to communicate with a local partner than a remote one, mobile users want to communicate and synchronize with nearby replicas. This requirement implies a peer replication model. Peer models face several challenges, but a particularly large one is scalability. Most replication systems are designed for scenarios that require only small numbers (fewer than a dozen) of writable replicas. However, mobile environments of the future may require many more replicas. Each mobile user requires a replica of shared data on his own machine, possibly increasing replication factors for some files into the low tens. In the near future, instead of each user having one stationary and one mobile machine, there will be many more \smart" devices capable of storing replicated data, such as palmtops, PDAs, and even watches. This trend could increase the number of replicas required by more than an order of magnitude. For maximum exibility and functionality, it must provide the ability for all replicas to generate updates, although some may never do so. Gray's(J.Gray , P.Helland, 1996) critique of the scalability of peer replication is based on transactional databases. Many systems requiring one item in the replication container must store the entire container. Mobile computers have much less disk storage available than desktop machines, especially desktops using remote file servers. A mobile replication system should replicate at a granularity that avoids storing unnecessary files(D. Ratner, G.Popek, 1996)

Mobile users prefer not to prearrange their mobility, so it is unreasonable to require them to perform registration before moving. Further, mobility cannot always be predicted or scheduled. We cannot require that users know a priori either when they will be-
come mobile or for how long.

## 6.1 The Replication Process (RP)

One of the central objectives of the usage of DBMS , on which most information systems are based, is to avoid unnecessary redundant data. Because the databases on mobile clients and/or database servers may become updated over time, replication must be a continuous process. In case of considering replication as a non-continuous process, the replica is a backup or an “useless” redundant copy and will not become updated but soon outdated.

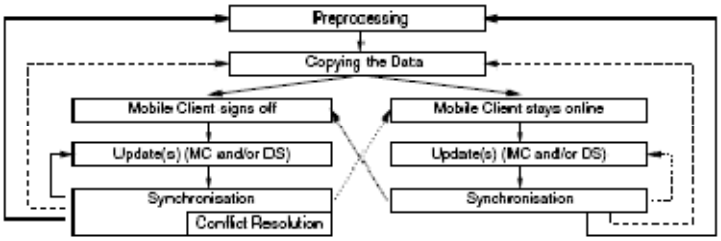


Figure 6: Replication process in mobile environment

As shown in Figure 6, the process is splitted into two parts. The first one describes the replication with mobile clients, which disconnect after getting the data. In the second part mobile units permanently stay online. In both cases the database on the DS must be preprocessed. The resulting replica set is subsequently copied to the mobile unit. After that, a modification of the data is possible on the database server as well as on the mobile client. Especially with off-line working mobile clients, updating causes inconsistent data.

This means that different versions of data may occur. Hence, mobile clients and database servers must synchronize with each other. Because of the high probability that off-line modified data on a mobile client conflicts with the version on the database server, advanced concepts for conflict resolution are necessary. After synchronization is finished, three ways are possible. First, a complete new replication can be started. In the second case, already existing preprocessing information are reused. In the third case, no new data is replicated. After synchronizing the updates, the mobile client signs off or stays online in the face of new updates.

### 6.1.1 Preprocessing

The preprocessing step of the RP is the definition of the data which shall be copied. This is necessary, because a complete copy of the data of the base station to the mobile host is not feasible in information systems with a huge amount of data. The replica must be only a part of the entire database.( Heuer,A., and Lubinski,A., 1998) presents an overview about data reduction techniques. Non-content-based data reduction approaches allow the reduction of the data without an user interaction. In commercial mobile database systems replica sets of data stored in the database are defined in such a manner. In contrast to non-content-based data reduction, content based data reduction allows to reduce the data in a more user dependent manner. Here the context of a user and/or of the users mobile equipment defines the replica set dynamically. Therefore the number of objects or the object size can be adapted by transforming queries or by transforming query results. For example, level of detail techniques (e.g. fisheye views) allow a format similar to classic replication in distributed databases e.g. semantically cached data may consist of tuples of one ore more tables in the relational case specific adaption of query results to the context of the inquiring hosts. The combined use of several data reduction techniques should be possible. Another aspect of the preprocessing step is to allow or deny updates on the replicas. This essential question corresponds directly to the definition of replica sets because the necessary conflict resolution in mobile environments must take the dis-connectivity of mobile units as well as the view update problem into account. Updateable replicas require a restrictive definition of replica sets. Furthermore, some more mobility dependent aspects must be considered in the preprocessing step. The usage of caches, which is well known from distributed database systems, improves the performance of information systems. Especially in mobile information systems caching strategies are used in a semantic manner. That means that the results of former queries are reused for faster answering new, semantical equivalent queries . The usability of those approaches for moving MCs is presented e.g. in (Ren, Q., and Dunham, M.H., 2000). In contrast to semantic caching, profiling techniques use explicit collected hints about users mobility and access patterns to predefine user specific replica sets. For example, in (S.-y. Wu and Y.-t. Chang. 1999) the location information are extracted from the cell where the MC is included in. If a MC tries to replicate data later on from the same DS, the history is used to pre-calculate possible relevant data.

### 6.2.2 Copying the Data

As described above, the second step in the RP is the delivery of the data. Here, the central problem is the connectivity between the mobile client and the base station. The kind of connection between base station and mobile client influences the choice of a transport media. If no network connection can be established, a disk-file-based transport has to be used. Because this method requires additional hardware like floppy drives, this file-based approach is not suitable for small mobile clients like PDAs. Therefore, a network connection must be available. For this purpose there are three approaches proposed in the literature: (1) Agent based replication, (2) Network protocol based replication and (3) Broadcast based replication. In the agent based replication approach (Pitoura, E., and Bhargava, B., 1995) application and database agents are used for the data transfer. Database agents are the interface to the database. They communicate with application agents using primitive application methods. These are predefined methods which allow the querying of a database and of agent data. The application agents transport the data to the target database agent, which then coordinates the necessary database operations on its site. The network protocol based replication allows the use of standard network protocols for the data transfer. Most commercial database management systems support the HTTP protocol. But also other protocol types like FTP or SMTP/POP are usable. This approach may also provide data in form of files. In systems with data, which is probably interesting for a great number of mobile clients, a broadcast approach can be used. In such scenarios the transport of the data can be done using a broadcast medium like radio waves. There are several publications which discuss the usability of broadcast techniques for the replication in mobile information systems (e.g. (Pedone,F., Guerraoui, R., and Schiper, A., 1998 ). But, the usage of broadcast techniques require either an additional up-link-chanel or a lot of additional functionality on the client site.

### 6.3.3 Updating the data and synchronization

One aim of the use of mobile equipment is the possibility of modifying the data at any place at any time. Here the problems of off-line updates will be discussed. It do not at best without querying the DS for separating needed information from the broadcast stream refer to the online case, because it is the classic and well known problem of updating data in distributed database systems. Here, normally quorum or primary copy approaches are used for keeping the database in a consistent state. If the updates were done off-line, it is not as easy to ensure that the consistency is given as in the online case. In fact, off-line modifications of the data cause differences between the database on the DS an the replicated data on the MC. This problem must be resolved in the synchronization step. There are two popular approaches for handling updates in mobile database systems. The first one is the use of tentative transactions (Gray, J., Helland, P., O'Neil, 1996). Here, all updates of mobile data are temporary updates and may be revoked in the synchronization step, if a given scope rule is violated. The second approach uses different versions of the snapshot for storing the data.As shown in (Shirish, H. Phatak and Badrinath, B.R,1996 ) this multi-version reconciliation approach works automatically for simple but not for complex conflicts. For real live problems both approaches require additional application logic. In the Bayou project (Demers, A., Petersen, K., 1994 ) the transaction based approach is complemented by per-write dependency checks and per-write merge procedures. These extensions allow the recognition and the automatic adjustment of conflicts, provided that the possibility of the appearance of these conflicts is known at the implementation time of the information system. Also, for advanced conflict handling in the multi-version reconciliation approach knowledge of possible conflicts is needed. As shown, the two steps "updating the data"

and ”synchronization” belong together and depend on the connectivity of the mobile clients. If data is modified online, the additional conflict resolution functionality is not necessary. The RP must react to this fact by adaptively choosing a conflict resolution level.

# 7. OPTIMISTIC REPLICATION IN MOBILE ENVIRONMENT

Optimistic replication is a group of techniques for sharing data efficiently in wide-area or mobile environments. The key feature that separates optimistic replication algorithms from their pessimistic counterparts is their approach to concurrency control. Pessimistic algorithms synchronously coordinate replicas during accesses and block the other users during an update. In contrast, optimistic algorithms let data be read or written without *a* priori synchronization, based on the .optimistic. assumption that problems will occur only rarely, if at all. Updates are propagated in the background, and occasional conflicts are fixed after they happen. It is not a new idea,1 but its use has exploded due to the proliferation of the Internet and mobile computing technologies. Optimistic algorithms offer many advantages over their pessimistic counterparts. First, they improve availability: applications make progress even when network links and sites are unreliable. Second, they are flexible with respect to networking, because techniques such as epidemic replication propagate operations reliably to all replicas, even when the communication graph is unknown and variable. Third, optimistic algorithms should be able to scale to a large number of replicas, because they require little synchronization among sites. Fourth, sites and users are highly autonomous: for example, services such as FTP and Usenet mirroring (Nakagawa 1996; Krasel 2000) let a replica be added with no change to existing sites. Optimistic replication also enables asynchronous collaboration between users, for instance in CVS([Cederqvist et al. 2001; Vesperman 2003) or Lotus Notes (Kawell et al. 1988). Finally, optimistic algorithms provide quick feedback, as they can apply updates tentatively as soon as they are submitted.

These benefits, however, come at a cost. Any distributed system faces a trade-off between availability and consistency (Fox and Brewer 1999; Yu and Vahdat 2002). Where a pessimistic algorithm waits, an optimistic one speculates. Optimistic replication faces the unique challenges of diverging replicas and conflicts between concurrent operations. It is thus applicable only for applications that can tolerate occasional conflicts and inconsistent data. Fortunately, in many real-world systems, especially file systems, conflicts are known to be rather rare, thanks to the data partitioning and access arbitration that naturally happen between users (Ousterhout et al. 1985; Baker et al. 1991; Vogels 1999;Wang et al. 2001).

# 8. HANDLING CONFLICTS IN OPTIMISTIC REPLICATION

Conflicts happen when some operations fail to satisfy their preconditions. Pessimistic algorithms prevent conflicts by blocking or aborting operation as necessary. Single-master systems avoid conflicts by accepting updates only at one site (but allow reads to happen anywhere). These approaches, however, come at the cost of lower availability. Conflicts can also be reduced, for example, by quickening propagation or by dividing objects into smaller independent units.

Some systems ignore conflicts: any potentially conflicting operation is simply overwritten by a newer operation. Such lost updates may not be an issue if the loss rate is negligible, or if users can voluntarily avoid lost updates. A distributed name service is an example, where only the

owner of a name may modify it, so avoiding lost updates is easy (Demers et al. 1987; Microsoft 2000).

The user experience is improved when a system can detect and signal conflicts. Conflict detection policies are also classified into syntactic and semantic policies. In systems with syntactic conflict detection policies, preconditions are not explicitly specified by the user or the application. They rely only on the timing of operation submission and conservatively declare a conflict between any two concurrent operations.

## 8.1 Detecting conflicts

An operation is in conflict when its precondition is unsatisfied, given the state of the replica after tentatively applying all operations before □in the current schedule. Conflict management involves two subtasks: detecting a conflict. Just like for scheduling, techniques range over the spectrum between syntactic and semantic approaches.

Syntactic conflict detection uses the happens-before relationship, or some approximation, to flag conflicts. That is, an operation is deemed in conflict when it is concurrent with another operation. Semantic approaches use knowledge of operation semantics to detect conflicts. In some systems, the conflict detection procedure is built in. For instance, in a replicated file system, creating two different files concurrently in the same directory is not a conflict, but updating the same regular file concurrently is a conflict (Ramsey and Csirmaz 2001; Kumar and Satyanarayanan 1993). Other systems, notably Bayou and IceCube, let the application or the user write explicit preconditions. This approach isolates the application-independent components of optimistic replication. e.g., operation propagation and commitment-from conflict detection and resolution. Semantic policies are strictly more expressive than syntactic counterparts, since one can easily write a semantic conflict detector that emulates a syntactic algorithm. Most operation-transfer systems use semantic conflict detector, mainly because the application already describes operations semantically- adding an application-specific precondition require little additional engineering effort. On the other hand, state-transfer systems use both approaches.

## 8.2 Resolving Conflicts

The role of conflict resolution is to rewrite or abort offending operations to remove suspected conflicts. Conflict resolution can be either manual or automatic. Manual conflict resolution simply excludes the offending operation from the common schedule and presents two versions of the object. It is up to the user to create a new, merged version and re-submit the operation.

**Automatic conflict resolution in file systems**. Automatic conflict resolution is performed by an application-specific procedure that takes two versions of an object and creates a new one. For instance, concurrent updates on a mail folder file can be resolved by computing the union of the messages from the two replicas. Concurrent updates to object (*.o) files can be resolved by recompiling from their source.

**Conflict resolution in Bayou**. Bayou supports multiple applications types by attaching an application-specific precondition (called the dependency check) and resolver (called the merge procedure) to each operation. Every time an operation is added to a schedule or its schedule ordering changes, Bayou runs the dependency check; if it fails, Bayou runs the merge procedure,

which can perform any fix-up necessary. For instance, if the operation is an appointment request, the dependency check might discover that the requested slot is not free any more; then the merge procedure could try a different time slot. To converge the state of replicas, every merge procedure must be completely deterministic, including its failure behavior (e.g., it may not succeed on some site and run out of memory on another).

# 9. APPLICATIONS OF OPTIMISTIC REPLICATION

Optimistic replication is particularly attractive in environments in which the communication between sites is slow and unreliable. Usenet, the wide-area bulletin board system deployed in 1979, is the oldest and still the most popular optimistically replicated service (Kurt Lidl, Josh Osborne, 1994). Usenet consists of thousands of servers connected in an ad-hoc way. In practice, each Usenet server stores only a subset of newsgroups to conserve the network bandwidth and the storage space..

Optimistic replication is also used to improve the performance and the availability of wide-area distributed data services, such as WWW ( Duane Wessels and Claffy K.,1997) FTP, and directory services (e.g., Xerox Clearinghouse(Alan J. Demers, Daniel H, `997) , DNS ( Mockapetris P.V. and Dunlap K. 1988 ), Active Directory, and Grapevine). Optimistic replication is attractive for them for many reasons. First, they must replicate data over unreliable longhaul network links. Second, they often operate under a tight budget. Finally, data inconsistency is inherent in these services (e.g., HTTP(Fielding R.,J.Gettus 1999) and DNS(Mockapetris P.V., 1987) explicitly allow stale data to be presented to users and using a replication algorithm with a loose consistency guarantee does not degrade their end-to-end service quality any more.

Optimistic replication is a key enabling technology in mobile computing systems that need to replicate data on portable devices. They allow users to read or update the data while they are disconnected, merging the modifications with other nodes when they reconnect. Applications with similar demands include mobile file systems (e.g., Coda(James J., Kistler, 1992) and mobile e-mail systems (e.g., Bayou (Douglas B. Terry, Maruin M., 1995) and Lotus Notes.

# 10. ADVANTAGES OF OPTIMISTIC REPLICATION

Optimistic replication algorithms own many advantages over pessimistic algorithms by letting replica contents diverge. The key advantage is their fault tolerance. Optimistic algorithms work well over slow or unreliable network links, because they can propagate updates among replicas in the background without blocking accesses to any replicas. This is in contrast with pessimistic counterparts that disallow accesses to replicas until it stores the newest content.

A related advantage is the networking flexibility. Optimistic algorithms work well over intermittent or incomplete network links(David H. Ratner, 1998, Karin Petersen, Mike J., 1997) by allowing updates to be exchanged between any pair of nodes. This property is essential in a mobile environment in which nodes can communicate with each other only occasionally and unpredictably. Some services deploy optimistic replication algorithms, because their fault tolerance and networking flexibility lets the system be built of inexpensive, failure-prone hardware and still maintain the consistency of data . Finally, optimistic algorithms often improve site autonomy by requiring less coordination among sites. For example, some services (e.g., FTP

mirroring ( Ikuo Nakagawa. Ftpmirror, 1996) allow a replica to be added without any administrative change on the existing replicas.

# 11. OPTIMISTIC REPLICATION CHALLENGES

Optimistic replication algorithms face two main challenges: keeping replicas consistent and scaling performance. While these challenges are not unique to optimistic replication, they are complicated, because these algorithms let updates be issued at multiple replicas at the same time and optimistic algorithms often discover update conflicts long after the updates are issued.

## 11.1 Maintaining replica consistency

Maintaining the consistency of replica, or controlling the quality of replica contents, is the most crucial function of any replication service. By definition, optimistic replication algorithms cannot guarantee strict single-copy consistency. Thus, the first challenge here is to define the type of replica consistency an algorithm can guarantee an implementation often follows naturally from the type of guarantee.

## 11.2 Eventual consistency

Eventual consistency demands the consistency of the replicas in a quiescent environment. In other words, it guarantees that whatever the current state of the replicas is, if no new update is issued and the replicas can communicate freely for a long enough period, the contents of all the replicas become identical eventually Eventual consistency is important for two reasons. First, it is the minimal requisite of a replication algorithm; without this guarantee, the replica contents may remain corrupted forever, making the system practically useless. Second, an eventually consistent service usually makes a best effort to disseminate updates quickly among replicas, and such a "best effort" is strong and practical enough for many applications. Several design choices exist as to how eventual consistency is achieved when updates may be issued concurrently — for example, whether to order updates totally or partially, or whether to let the user specify the ordering or to let the system determine the ordering automatically.

## 11.3 View consistency

Eventual consistency by itself provides little end-to-end guarantee about the quality of data — that the replica contents eventually converge give means little to users. The goal of view consistency guarantees is actually to control the quality of data by intervening on replica read (and sometimes update) requests. This guarantee is optional, and in fact, many systems serve arbitrarily stale data to users. View consistency includes causal consistency that preserves partial orderings among read and write requests, and bounded inconsistency that explicitly limits the degree of replica inconsistency. For example of causal consistency, consider a replicated password database(Terry D. B. , Demers A.J., 1994). A user may change her password on one replica and later fail to log in from another replica using the new password, because the change has not reached the latter replica. Such a problem can be avoided by having a causal read guarantee, that is, to make a read request honor the past updates made by the same user.

### 11.4 Scalability

Optimistic replication algorithms must perform efficiently under a large workload to be practically useful. The size of the workload grows along three axes: the object size, the number of replicas, and the number of objects. These axes are mostly orthogonal — for example, some algorithms scale to many replicas well but not to large objects.

The space efficiency of a system is often as important as the performance. Replication algorithms must maintain several data structures as well as the object contents themselves — the log of updates, the names of other replicas, etc. These data structures should be small, and should be able to be trimmed if necessary so as to leave the maximum amount of space for what users actually want, i.e., the object contents

## 12 CONCLUSIONS

In this paper a brief summary of replicated data management strategies in the traditional distributed database environment and a generalized replication process in mobile environment have been discussed. Management of data in the mobile computing environment offers new challenging problems. Existing DBMS software need to be upgraded to adapt them to the new environment. To be able to do so, the critical parameters need to be understood and defined. Mobility brings in new dimension to the existing solutions to the problems in distributed databases. This paper surveyed some of the problems and existing solutions of database software in that direction. Replicating data at many nodes while node has been disconnected and letting anyone update the data is problematic. For concurrency control optimistic replication is one of the solutions of replicating data in mobile environment. Optimistic replication faces the unique challenges of diverging replicas and conflicts between concurrent operations. The challenges and conflicts may be faced during optimistic replication in mobile transaction were highlighted so that while mobile transaction such problems would be considered.

## REFERENCES

[1] Ashraf, Ahamad, " A new Optimistic Replication Strategy for Large Scale Mobile Distributed database Systems". International Journal of Database Management Systems(IJDMS). Vol.2 No.4, November, 2010.

[2] Abawajy, Deris, J., Omer, M., "Anovel data replication and management protocol for mobile computing systems". Mobile Information Systems, 2(1): 3-19, IOS Press, Netherlands, 2006.

[3] Alan J. Demers, Daniel H. Greene, Carl Hauser, WesIrish, and John Larson. Epidemic algorithms for replicated database maintenance. In 16th ACM Symp. on Princ.of Distr. Computing (PODC), pages 1–12, Santa Barbara, CA, August 1997.

[4] Alastair Wolman, Geoff Voelker, Nitin Sharma, Neil Cardwell, Anna Karlin, and Henry Levy. On the scale and performance of cooperative Web proxy caching. In 17t h Symp. on Operating Systems Principles (SOSP), Kiawah Island, SC, December 1999.

[5] Bernstein, P. "Principles of Transaction Processing". Morgan Kaufmann Publisher Inc., 1997.

[6] Bernstein, A. Hadzilacos, P. V., Goodman N.Concurrency Control and Recovery in Database Systems. 1987. Microsoft Research. URL: http://research.microsoft.com/en-us/people/philbe/chapter8.pdf.

[7] Cederqvist, P., Pesch, R., ET AL. 2001. Version management with CVS. http://www.cvshome.org/docs/manual.

[8] Douglas B. Terry, Marvin M. Theimer, Karin Petersen, Alan J. Demers, Mike J. Spreitzer, and Carl H. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In 15th Symp. on Operating Systems Principles (SOSP), pages 172–183, Copper Mountain, CO, December 1995.

[9] David H. Ratner. Roam: A Scalable Replication System for Mobile and Distributed Computing. PhD thesis, UC Los Angeles, 1998. UCLA-CSD-970044

[10] Duane Wessels and K. Claffy. RFC2186: Internet cache protocol. http://info.internet.isi.edu/in-notes/rfc/- files/rfc2186.txt, September 1997.

[11] Elmasri and Navathe, Fundamentals of database Systems, Addison Wesley, New York, 2004.

[12] Fox, A. and Brewer, E. A. 1999. Harvest, Yield, and Scalable Tolerant Systems. In 6th Workshop on Hot Topics in Operating Systems (HOTOS-VI) (Rio Rico, AZ, USA, March 1999), pp. 174.178.

[13] Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., and Berners-Lee T.. RFC2616: Hypertext transfer protocol – HTTP/1.1. http://info.internet.isi.edu/innotes/ rfc/files/rfc2616.txt, June 1999.

[14] Gray, J Helland, P. and O'Neil, P "The dangers of replication and a solution". In Proceedings of the ACM SIGMOD International Conference on management of Data, pp. 173-182, 1996.

[15] Helal, A. Heddaya, a. and Bhargava, B. "Replication Techniques in Distributed systems". Kluwer Academic Publisher, 1996

[16] Ikuo Nakagawa. Ftpmirror – mirroring directory hierarchy with ftp. http://noc.intec.co.jp/ftpmirror.html, November 1996.

[17] James Gwertzman and Margo Seltzer. World-wide web cache consistency. In USENIX Winter Technical Conference, February 1996.

[18] James J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda file system. ACM Trans. on Computer Systems (TOCS), 10(5), February 1992.

[19] Kurt Lidl, Josh Osborne, and Joseph Malcolm. Drinking from the firehose: Multicast USENET news. In USENIX Winter Technical Conference, 1994. ftp://ftp.uu.net/networking/news/muse/usenixmuse. ps.gz.

[20] Kawell, L., JR., BECKHART, S., HALVORSEN, T., OZZIE, R., AND GREIF, I. 1988. Replicated document management in a group communication system. In Conf. on Comp.-Supported Coop. Work (CSCW) (Chapel Hill, NC, USA, October 1988).

[21] Karin Petersen, Mike J. Spreitzer, Douglas B. Terry, Marvin M. Theimer, and Alan J. Demers. Flexible Update Propagation for Weakly Consistent Replication. In 16th Symp. on Operating Systems Principles (SOSP), pages 288–301, St. Malo, France, October 1997

[22] Monterio, J Brayner, A. and Lifschitz, S. "A mechanism for replicated data consistency in mobile computing environments".In proceedings of the ACM symposium on Applied computing . Seoul, Korea,pp.914-919,2007

[23] MICROSOFT. 2000. Windows 2000 Server: Distributed Systems Guide, Chapter 6, pp. 299- 340. Microsoft Press, Redmond, WA, USA.

[24] Mockapetris P.V..RFC1035:Domain names — implementation and specification. http://info.internet.isi.edu/innotes/ rfc/files/rfc1035.txt, November 1987.

[25] Mockapetris P.V. and Dunlap K.. Development of the domain name system. In ACM SIGCOMM, Stanford, CA, August 1988.

[26] Nakagawa, I. 1996. FTPmirror . mirroring directory hierarchy with FTP. http://noc.intec. co.jp/ftpmirror.html.

[27] Oracle 2009a. Oracle Database Advanced Replication 11g Release 2 (11.2) URL: http://download.oracle.com/docs/cd/E11882_01/server.112/e10706/toc.htm.

[28] Ramakrishnan, R & Gehrke, J. Database Management Systems. 2nd edition 2002. Mcgraw-Hill Thomas, O., McLean, I. Optimizing and Maintaining a Database Administration Solution by Using SQL Server 2005. 2006. Microsoft Press.

[29] Ratner, D. Reiher, P. and Popek, G. "Roam: a scalable replication system for mobility". Mobile Networks and Applications, 9(5):537-544,2004

[30] Ratner, D. Reiher,P. Popek,,G. and Kuenning,G. "Replication Requirements in mobile environments". Mobile Networks and Applications,6(6):525-533,2001

[31] Ratner, D., Popek, G., and Reiher, P., Peer replication with selective control. UCLA Computer Science Department Technical Report CSD-960031, July 1996.

[32] Rennhackkamp, M., "Mobile Database Replications , Concepts, Configurations and Concerns related to Managing Replication", DBMS, Vol.10(11),pp 81-84,2001.

[33] Ramsey, N. and Csirmaz, E. 2001. An Algebraic Approach to File Synchronization. In 9th Int. Symp. on the Foundations of Softw. Eng. (FSE) (Austria, September 2001).

[34] Tolia, N. Satyanarayan, M. and Wolbach,A. "Improving mobile database access over widearea networks without degrading consistency". Proceedings of the 5th International Conference on Mobile systems ,applications and services, San Juan , Puerto Rico, pp.71-84, 2007

[35] Thomas, O., McLean, I. Optimizing and Maintaining a Database Administration Solution by Using SQL Server 2005. 2006. Microsoft Press.

[36] Terry D.B., Demers A.J., Petersen K., Spreitzer M.J., Theimer M.M, and Welch B.B. Session guarantees for weakly consistent replicated data. In Proceedings International Conference on Parallel and Distributed Information Systems (PDIS), pages 140–149, Austin, TX, September 1994.

[37] Wiesmann, M., Pedone, F., Schiper, A., Kemme, B., Alonso, G. 2000. Understanding Replica-tion in Database and Distributed Systems. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.9166&rep=rep1&type=pdf.

[38] Weismann, M. Pedone,F., Schiper, A., Kemme,B. and Alonso,G., "Database Replication Techniques: A Three parameter Classification", Proceedings of the 19th IEEE Symposium. Reliable Distributed Systems, pp 202, 2000

## Author

She is working as Assistant Professor in IT Dept, running successive 16th year in Institute of Technology & Science, Ghaziabad with one year in a software development industry. Authored 4 book/workbooks on computer science for MCA/B Tech/ BCA programmes. Besides this published and presented more than 20 papers in various national/ international journals and conferences