A New Optimistic Replication Strategy for Large-scale Mobile Distributed Database Systems

Ashraf Ahmed[1], P.D.D.Dominic [2], Azween Abdullah [2] and Hamidah Ibrahim [3]

[1]Department of Computer, University of Khartoum, Khartoum, Sudan
ashrafafadel@hotmail.com
[2]Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Perak, Malaysia
{dhanapal_d,Azweenabdullah}@petronas.com.my
[3]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Selangor, Malaysia
hamidah@fsktm.upm.edu.my

ABSTRACT

This paper proposes a new optimistic replication strategy for maintaining the eventual consistency in large-scale mobile distributed database systems. The proposed strategy consists of three components in order to support the characteristics of such systems. These components are: replication architecture, updates propagation protocol, and replication method. The purpose of the replication architecture is to provide a comprehensive infrastructure for distributing replicas among wide areas. The purpose of the propagation protocol is to transfer data updates between the components of the replication architecture in a manner that achieves the eventual consistency of data and improves the availability of recent updates to all replicas. The replication method provides a mechanism for implementing the propagation protocol, and automating the propagation of updates among the different replicas. The effectiveness of the proposed strategy is compared with two baseline replication strategies and shown that it achieves updates propagation delay reduction. Also, the results showed that the horizontal extension provided by the proposed strategy is more suitable than the vertical extension for large scale mobile distributed database systems.

KEYWORDS

Optimistic Replication, Eventual Consistency, Updates Propagation, Replication Method

1. INTRODUCTION

Rapid advancements in wireless technologies and portable devices have given mobile computing considerable attention in the past few years as a new dimension in data communication and processing, and a fertile area of work for researchers in the areas of database and data management [3,5]. As mobile computing devices (e.g. laptop, PDA, and cell phones) become more and more common, mobile databases are becoming popular [2,20]. According to [13] and [22], the mobile database can be defined as a database that is portable and physically separate from a centralized database server but is capable of communicating with server from remote sites allowing the sharing of data.

Mobility of users and portability of devices pose new problems in the management of data [10,19,24], including transaction management, query processing, and data replication. Therefore, mobile computing environments require data management approaches that are able to provide complete and highly available access to shared data at any time from any where. One way to achieve such goal is through data replication techniques. The importance of such techniques is increasing as collaboration through wide-area and mobile networks becomes popular [30]. However, maintaining the consistency of replicated data among all replicas represents a challenge in mobile computing environments when updates are allowed at any replica.

This paper addresses the problem of maintaining consistency of replicated data in Large Scale Mobile Distributed Database Systems (LMDDBSs). This type of systems is characterized by a large number of replicas (i.e. thousands of replicas) and a large number of updates (i.e. tens of updates may be performed on each replica at any period of time) that are performed on these replicas. The replicas are distributed over both mobile and fixed environments. Examples of such systems include mobile health care, mobile data warehousing, news gathering, and traffic control management systems. The characteristics of LMDDBSs, and the ability of the mobile users to issue a large number of concurrent update operations, which may occur during the disconnection periods extremely aggravate this problem than in the ordinary replicated database systems, which they have a small number of replicas. The concurrent updates during disconnection periods influence consistency of the replicated data by leading to a divergence in the database states (i.e. the data that are stored in the database at a particular moment in time) within replicas that are hosted in mobile environments, and between these replicas and the replicas that are hosted in fixed environments.

To cope with the problem of maintaining consistency of the replicated data, several replication strategies are proposed. These strategies are divided into optimistic and pessimistic approaches [15,30].

Pessimistic strategies operate under a pessimistic assumption that if an inconsistency can occur, it will occur. Therefore, these strategies restrict updates to a single replica or a group of replicas by locking access to these replicas during the processing of update requests. Then, updates are applied to all other replicas. Accordingly, the pessimistic strategies provide the strong consistency guarantee that is called one-copy equivalence [26]. This consistency guarantee requires that all replicas be mutually consistent (i.e. have identical values for all shared data items) at the end of each update operation. Examples of such strategies include ROWA [1,18], and primary copy approaches [16,29]. According to the description of the pessimistic strategies, it can generally be said that these strategies rely on two factors as follows. The first factor is a reliable and constant communication between replicas. The second factor is the coordination between the replicas that are involved in the performing of operations, which may require the blocking of access to these replicas during performing the operations. However, these factors are not feasible in mobile environments where frequent disconnections is common, which makes these pessimistic strategies are not suitable for maintaining consistency of replicated data in such environments, especially when some of mobile nodes hold updatable replicas.

In contrast with pessimistic strategies, optimistic strategies operate under the optimistic assumption that inconsistencies, even if possible, rarely occur [30]. Operations (read and write) may be executed on any replica. At the reconnection time, the system must first detect inconsistencies and then resolve them. Accordingly, optimistic replication allows replicas to access and update data independent from one another by delaying consistency checks. Then, replicas exchange updates with one another in a process called reconciliation, which occurred periodically [8,23]. During the reconciliation process, two replicas exchange all updates that occurred since the last reconciliation.

Optimistic replication is often used to increase database availability in systems where communication is unreliable or nodes require access to data while disconnected from the network (e.g. mobile environments) [15]. However, optimistic replication strategies face the challenge of keeping replicas consistent. This challenge is complicated, because these strategies let updates to be issued at multiple replicas at the same time. Accordingly, optimistic replication strategies cannot guarantee strong consistency through achieving one-copy equivalence. Instead of that, they provide a weak type of replica consistency guarantee called eventual consistency [11,30]. Eventual consistency guarantees that the contents of all the replicas become identical eventually. Eventual consistency is important because it is the minimal requisite of a replication strategy; without this guarantee, the replica contents may remain corrupted forever, making the system practically useless. To achieve eventual consistency, optimistic replication algorithms should provide mechanisms for rapid propagation (dissemination) of updates among replicas in order to minimize the divergence between them.

Optimistic replication is used in several solutions for handling data replication issues in mobile environments. This is because it meets the goal of providing higher availability. However, these strategies have not explicitly addressed the issues of consistency and availability of data in large scale mobile distributed database systems that contain large number of both fixed and mobile hosts.

Therefore, this paper comes to a conclusion that additional research toward a dedicated replication strategy for LMDDBSs is needed to investigate and address data consistency issue in such systems.

Accordingly, the paper proposes a new replication strategy that acts in accordance with the characteristics of these systems (i.e. large number of updatable replicas).

This paper is organized as follows. The next section provides the related work. Section 3 outlines the proposed replication strategy, and presents the system model that is used in this paper. Section 4 describes the replication architecture. Section 5 gives the details of the updates propagation protocol and its performance evaluation. Section 6 describes the replication method. Section 7 concludes the paper.

2. RELATED WORK

Using optimistic replication in mobile environments has been studied in several research efforts. Roam [7,8] is an optimistic replication system that provides a scalable replication solution for the mobile user. ROAM is based on the Ward Model [6]. Replicas are grouped into wards (wide area replication domains). All ward members are peers, allowing any pair of ward members to directly synchronize and communicate. Each ward has a ward master that maintains consistency with the other wards. Updates are exchanged within each ward (i.e. between ward members) and among wards (i.e. between ward masters) using ring topology. Accordingly, Roam employs optimistic replica control mechanism that ensures an eventual convergence for replica updates to maintain the consistency within each ward and among wards. ROAM tries to provide high scalability without discussing a mechanism of ensuring rapid propagation of large numbers of updates that can be performed on both fixed and mobile replicas, which are distributed over wide geographic areas.

An n-ary tree based updates propagation strategy [25,27,28] assumed an environment where update information is immediately sent to all peers holding replicas when an update occurs. The proposed strategy creates an n-ary tree, whose root is the owner of the original data while the other nodes are peers holding its replicas, and propagates the update information according to the tree. Each peer in the tree records its parent and children, and by using this information, the location of a newly participating peer in the tree is autonomously determined. However, in this strategy, the updated data must be propagated to all replica holders and this causes heavy traffic for update propagation when the number of replicas becomes large. Also, there is an overhead originated from the need of each peer to manage information about the parent and children peers.

A multi-master scheme is used in [14], that is, read-any/write-any. To reach an eventual consistency in which the servers converge to an identical copy, an adaptation in the primary commit scheme is used. In this adaptation, a server chosen as a primary has the responsibility to synchronize and commit the updates. The committed updates are propagated to the other servers. This strategy inherits the drawbacks of primary-copy approach, since it relies on a selected server that is responsible for synchronizing all updates between the different replicas.

A hybrid replication strategy is presented in [12] that has different ways of replicating and managing data on fixed and mobile networks. In the fixed network, the data object is replicated synchronously to all sites in a manner of logical three dimensional grid structure, while in the mobile network, the data object is replicated asynchronously at only one site based on the most frequently visited site. In this strategy, the synchronous replication hinders the fixed network to be scale to wide areas.

Cedar [17] is a replication strategy focuses on preserving eventual consistency through using a simple Client/Server design in which a central server holds the master copy of the database. At infrequent intervals when a client has connectivity to the server (which may occur hours or days apart), its replica is refreshed from the master copy. The using of a central server to hold the master copy in this strategy, limits its implementation in large scale environments.

A mobile database replication strategy called Transaction-Level Result-Set Propagation is proposed in [9]. Each fixed and mobile nodes store a replica of the data. The mobile node is allowed to update its local replica. However, updates locally committed at the mobile nodes need to be verified at the fixed node before they can be globally committed.

In summary, we argue that existing replication strategies are not coping well with the characteristics of large-scale mobile systems containing large number of geographically distant replicas that experience large number of updates. Accordingly, such systems demand new solutions for addressing data consistency through ensuring fast propagation of recent updates among replicas as well as supporting

scalability for encompassing new replicas, especially when the replication system covers new geographic areas.

3. THE PROPOSED REPLICATION STRATEGY

The proposed replication strategy encompasses three components: replication architecture, updates propagation protocol, and replication method. The purpose of the replication architecture is to provide a comprehensive infrastructure for distributing replicas among wide areas, improving data availability, and supporting large number of replicas in mobile environments by determining the required components that are involved in the replication process. The purpose of the propagation protocol is to transfer data updates between the components of the replication architecture in a manner that achieves the consistency of data and improves the availability of recent updates to all replicas. The replication method provides a mechanism for implementing the propagation protocol, and automating the propagation of updates among the different levels of the replication architecture.

3.1. System Model

This paper considers a large-scale environment that consists of Fixed Hosts (FH), Mobile Hosts (MH), a replica manager on each host, and a replicated database on each host. A part of fixed hosts represent servers with more storage and processing capabilities than the rest. The replica manger is a software that manages and facilitates the application's accesses (read and update operations) to the replicated database. A replicated database is called mobile database when it is stored on a mobile host. The replicated database contains a set of objects stored on the set of hosts. The database is fully replicated on the servers, while it is partially replicated on both fixed and mobile hosts. Update can take place at any host. Update information is sent to the other hosts in a form of message. A replica is in consistent state if and only if all updates that are performed on the other replicas are received by this replica. Each host has a unique address, called *Host- ID*. The information of hosts and their replicated data are stored on objects called *Hosts-Obj* and *Host-Replicated-Objects-Obj*, respectively, which are replicated on each server. When a failure occurs in any host, the affected host is considered as disconnected. In this paper, the terms *replica* and *host* will be used interchangeably because each host has a replica.

4. REPLICATION ARCHITECTURE

The proposed replication architecture (see Figure 1) considers a total geographic area called the master area that has a server called Master Server (MS) and a set of fixed hosts. The master area is divided into a set $Z = \{z_1, \ldots, z_n\}$ of zones. Each zone has a server called Zone Server (ZS) and a set of fixed hosts and it consists of a set $C = \{c_1,\ldots, c_m\}$ of smaller areas called cells. Each cell represents an area, where the mobile users can perform their duties at a particular period of time before moving to another cell. Each cell has a server called Cell Server (CS). In this architecture, the network is divided into fixed network and mobile network. The fixed network consists of fixed hosts and wired local area network to connect them in the master area. Also, it includes wide area network to connect the master server with zone servers, and to connect zone server with underlying cell servers. The cell server is augmented with a wireless interface and acts as a mobile support station for connecting mobile hosts to the fixed network. On the other hand, the mobile network consists of wireless network and mobile hosts in the cell area.

To provide more flexibility and application areas for this architecture, replicas are divided into three levels:

Master Level: In this level, the replica that is stored on the master server must be synchronized with replicas from the zone level. The master server is responsible for synchronizing all changes that have been performed on both infrequently changed data and frequently changed data with the lower level.

Zone Level: In this level, each replica must be synchronized with replicas from the lower level. The zone server is responsible for synchronizing all intra-level changes with the master server.

Cell Level: Each replica in this level is updated frequently, and then synchronized with the cell server's replica and in turn the cell server synchronizes all intra-level data with the zone server.

In this architecture, initially, the database is stored on the master server. When dividing the master area into multiple zones, a replica of that database is distributed to zone servers. Similarly, new replicas are

created when dividing the zone area into multiple cells to represent the cell servers and when registering new mobile hosts and fixed hosts in the replication system. The information of each new replica is stored on the hosts object in the server of the area where the replica is created and then it is replicated to other servers. This information includes the *Host ID*, *Host Type* (FH, MH, CS, ZS, and MS), and *Region* where it is registered. The details of the replicated objects on that host are stored on the object of host's replicated data.

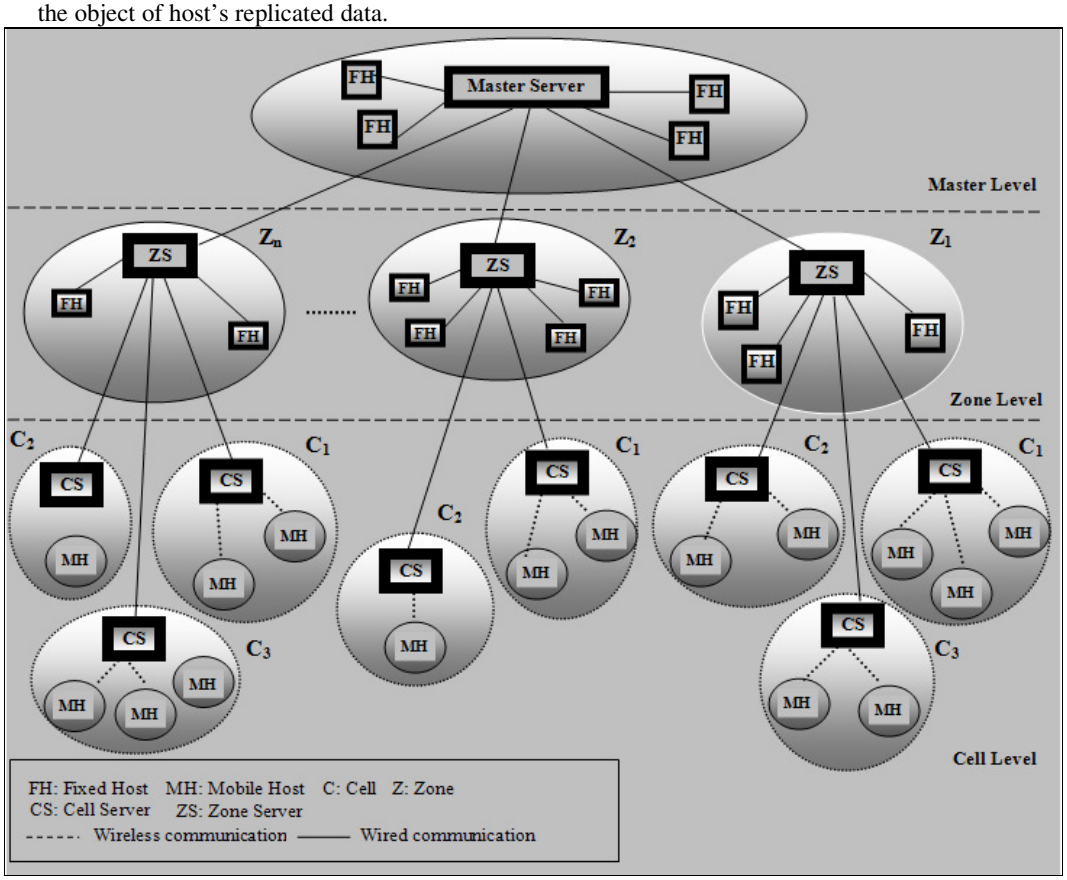

Figure 1. The replication architecture for large-scale mobile environments

5. WHEEL-BASED UPDATES PROPAGATION PROTOCOL

This section provides the details of the proposed protocol for updates propagation through the components of replication architecture. The protocol consists of a logical structure for arranging replicas and propagation mechanisms for exchanging updates among these replicas. The logical structure is a wheel-like structure that organizes replicas according to their types, areas where they inhabit (cell, zone, and master areas), and responsibility with regard to updates propagation. The propagation mechanisms act as interaction mechanisms between the replicas for propagating recent updates from their sources to other replicas that are distributed over the wheel. Accordingly, the resulting protocol is called Wheel-Based updates propagation protocol.

5.1 Updates Propagation Wheel

The logical structure that is involved in the updates propagation is a water wheel inspired structure called updates propagation wheel, which represents a logical structure for exchanging recent updates between the hosts that are distributed over the replication architecture.

The application of the water wheel structure here is arising from its general design and functionality. The water wheel structure [21] links an axle (i.e. acts as a central point) with multiple buckets (act as points) that are located in different directions on a circular rim through spokes. The functionality of the water wheel depends on the rotation of the buckets that are located on the rim after they are filled by

the water. This rotation leads to the revolution of the whole wheel including the center point. To apply this idea, updates propagation wheel is structured in a manner that includes the basic components of the water wheel with different explanations and functionalities.

Given N replicas of the database, the propagation protocol organizes them logically into the wheel structure based on their areas and types as shown in Figure 2. The following definition will formally define the propagation wheel (PW).

Definition 1. PW is 9-tuple < F, H, S, L, R, P, U, M, T >, where:
$F= \{FH_1,\ldots, FH_n\}$ is a finite set of fixed hosts that act as fixed points (i.e. buckets), which are distributed over the wheel.
$H= \{MH_1,\ldots, MH_h\}$ is a finite set of mobile hosts that act as mobile buckets over the wheel.
$S= \{s_1,\ldots, s_s\}$ is a finite set of servers that act as fixed center points where a sub set of F, H, or S can be connected to each center point.
$L = \{l_1,\ldots, l_l\}$ is a finite set of communication links that act as spokes for linking the different points distributed over the wheel.
$R = \{r_1, r_2, r_3\}$ is a finite set of virtual circular rims that act as a collection of points that have same area.
$P= \{p_1, \ldots, p_p\}$ is a finite set of parts that constitute each rim. Each part is called a sector.
$U: F \cup H \cup S \rightarrow \{1, 2, 3,\ldots, k\}$ is a function for assigning a unique identifier serially for each host in the wheel according to its type.
$M = \{m_1, m_2, m_3\}$ is a finite set of mechanisms for exchanging updates between the different points in the wheel.
$T= \{t_1,\ldots, t_{n+h+s}\}$ is a finite set of total number of updates that are currently stored in each host

Center points. As depicted in Figure 2, the different types of hosts are represented by circles in the propagation wheel. Some hosts act as center points where multiple spokes are collected on them. These points represent the servers of the different areas. Accordingly, these points can be classified into master server, zone servers, and cell servers according to their areas. The master server acts as the main center point, while both zone servers and cell servers act as secondary center points. The center points are linked through spokes to a set of either other center points or ordinary points (i.e. points act as either fixed or mobile hosts) in the next outer rim.

In this wheel, both center and ordinary points represent the different types of hosts of the replicated system, while the spokes between them represent the network connections (channels).

Rims. They are formed by the hosts that have same area regardless of their directions. Accordingly, we have three rims as follows.

(i) Master Rim: It contains all zone servers as well as fixed hosts on the master area. The master server is responsible for all hosts exist in this rim in that it receives their updates and sends their missed updates to them.
(ii) Zone Rim: It contains all cell servers as well as fixed hosts on the zone area. The zone server is responsible for a part of this rim called a sector, which represents the cell servers and fixed hosts that are located in its area.
(iii) Cell Rim: It contains all mobile hosts and fixed hosts in the cell area. The cell server is responsible for a part of the cell rim which represents the fixed hosts and mobile hosts that are located in its cell.

The rotation of the MHs in both clockwise and anticlockwise directions in the cell rim can be envisioned as a motivation for the revolution of the wheel since the MHs are located here on the most outer rim (i.e. cell rim).

Sectors. Both the zone and cell rims have multiple sectors (i.e. they are divided into multiple parts). Each sector consists of a set of hosts that have same area (either zone or cell) and are connected to the same center point in the next inner rim (i.e. their area's server). For example, the fixed hosts and cell servers that belong to specific zone form a sector on the zone rim and they connect to the server of this zone in the master rim.

The sector is named using the name of the responsible secondary point in the next inner rim. For example, the sector S_{Z2}-$Sec= \{CS_{21},\ldots, CS_{2c}\} \cup \{FH_{21},\ldots, FH_{2v}\}$ represents a part of the zone rim under the responsibility of the zone server number 2.

Spokes. The hosts in a given rim are linked to their related hosts in another rim or nearby hosts in the same rim through spokes. Two categories of spokes exist in the propagation wheel as follows.

a. **Fixed spokes.** This category links the servers in a given rim with their related servers and fixed hosts in the next outer rim.
b. **Temporary spokes.** They link the cell server with mobile hosts that are currently roaming in its cell (i.e. its sector). Also, this category links two nearby hosts from the same type in same level. For example, it links two nearby cell serves in the same zone or two mobile hosts in the same cell.

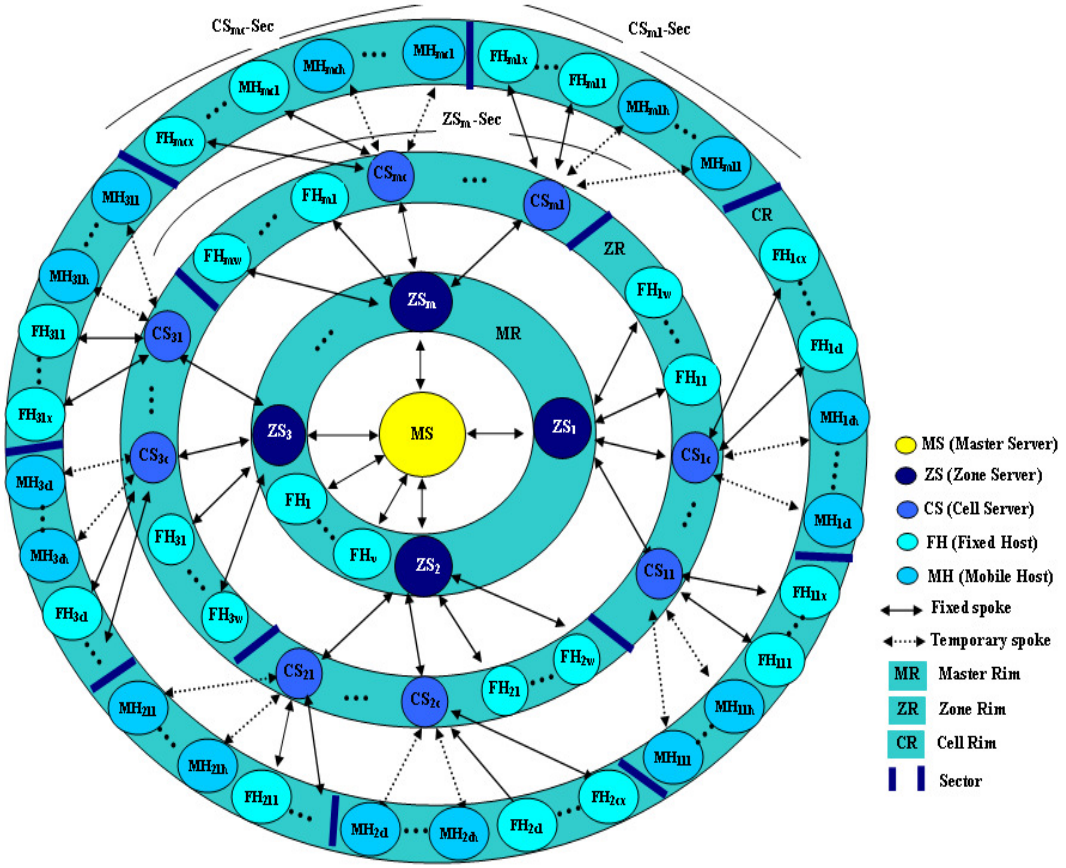

Figure 2. Updates propagation wheel

Naming schema. The hosts are named using the schema: $Host\text{-}Type_{\text{Zone-No Cell-N0 Host-Serial}}$ (e.g. FH_{212} is the name of the fixed host number 2 in cell number 1, which belongs to zone number 2). MHs are named by considering the zone and cell areas where they have been registered at the first time. The cell servers are named using the following schema $CS_{\text{Zone-No Cell-Serial}}$ (e.g. CS_{41} is the name of the cell server number 1 in zone number 4). The zone servers are identified serially.

Propagation mechanisms. Three basic mechanisms are identified for propagating updates from their sources to a set of other hosts in the propagation wheel as follows.

1. **Outer-to-Inner Propagation.** In this mechanism, updates flow through the rims in the direction of the wheel center from their sources in an outer rim into an inner rim until they pour into the master center point. Each intermediate rim keeps the poured updates for a certain period for the purpose of accumulating them before pouring them into the next inner rim. Accordingly, the steps that are carried out for this type of propagation are as follows.
   - Updates on the hosts (i.e. MHs and FHs) that populate the cell rim flow into their responsible secondary center points (i.e. CSs) in the zone rim.
   - The secondary center points in the zone rim accumulate the poured updates from the cell rim for further processing that implies the ordering of these updates.

- Processed updates on the CSs of zone rim flow into their responsible secondary center points (i.e. ZSs) that populate the master rim.
- The secondary center points in the master rim accumulate the poured updates from the zone rim for processing them in a total manner.
- All accumulated and processed updates on the zone rim flow to the master center point.

This type models the propagation of updates from the lowest level in the replication architecture to the highest level. The lowest level represents the cell level, which is modeled by the cell rim, while the highest level represents the master server and it is modeled by the main center point. Accordingly, this mechanism can be called Bottom-Up propagation (BUP).

2. **Inner-to-Outer Propagation.** In this mechanism, totally ordered updates by the main center point are pumped from an inner rim into an outer rim in the direction of the most outer rim. Each intermediate rim contributes the pumping by pushing those updates to reach the most outer rim. Accordingly, the steps that are carried out for this type of propagation are as follows.
   - Totally processed updates on the main center point are pumped into the secondary center points that populate the master rim.
   - Each secondary center point in the master rim pushes those updates to its underlying secondary center points that populate the zone rim.
   - Each center point in the zone rim pushes those updates to underlying points that populate the cell rim.

   This type models the propagation of updates from the highest level (i.e. master level) in the replication architecture to the lowest level (i.e. cell level). Thus, this mechanism can be called Top-Down propagation (TDP).

3. **Inside-Sector propagation.** In this propagation, updates are exchanged inside the rim between the hosts that have same type and sector (i.e. they populate same area), and have a possibility to communicate with each other. Accordingly, this mechanism is also called P2P propagation. Each peer pumps its received updates (either from other rim or generated on it) into a subset of the other peers that can communicate directly with them in the same sector. The pumping process continues from the other peers to ensure that updates are pushed to all peers in the sector. In the case of existing of more than one master area, this implies exchanging of updates between the master servers of the wheels that represent these master areas in a peer-to-peer manner. This is because there is no higher level than the master server.

The three basic propagation mechanisms represent the possible alternatives for exchanging recent updates between the different types of hosts that are included in the propagation wheel. Outer-to-Inner propagation is necessary to ensure that all recent updates, which occurred in the lower level hosts reach the servers that are located in the higher levels. This is because these servers cover different areas, and are required to provide their underlying hosts with the recent updates that occurred in the other areas. Inner-to-Outer propagation is required for each server to disseminate the received updates from other areas to the hosts that are located in its area. Inside-Sector propagation mechanism is provided to enable the hosts, which exist in the same area to exchange their updates directly without needing to send these updates to the server of area where they are located. For example, some hosts may be involved in performing duties that are associated with their area. Accordingly, they can share their updates using this mechanism without needing to send these updates to the other areas.

5.2 Hybrid Propagation Mechanisms

The following mechanisms act as a hybrid of two or all basic mechanisms for propagating updates from their sources to all hosts:

1. **Hierarchical-Concentrate propagation.** It represents a hybrid of both Outer-to-Inner and Inner-to-Outer propagation mechanisms. In this mechanism, updates are propagated to all hosts by delegating the responsibility of the propagation to the center points, which have a stable connectivity with their both underlying center points in the next outer rim and the responsible center point in the next inner rim. For instance, the main center point, which represents the server that exists in the highest level (i.e. master server) in the replication architecture, has a stable connectivity with the servers that cover all areas in the replication system (i.e. zone servers). The resolution of updates conflicts through updates ordering process is carried out at the server in the higher level. The steps are as follows.
   - The hosts in the lower levels propagate their updates using BUP mechanism to the server in the higher level till they reach the server in the highest level.

- The collected updates are propagated from the highest level to the lower levels using TDP mechanism.

2. **P2P-Concentrate propagation.** It represents a hybrid of the three basic mechanisms for exchanging updates in the same area (i.e. same cell, same zone, or same master area). In this hybrid, the role of the server of the area where peers inhabit (i.e. the center point in the next inner rim) is eliminated to allow the peers to exchange their updates without needing to send them to the higher level. However, peers need to propagate their updates to this server when these updates should be propagated to the other areas of the replication architecture. The steps are as follows.
   - The lower level hosts propagates their updates to the servers in the higher level of their region using BUP propagation.
   - Each server propagates those updates to the peers that can communicate with them until updates reach all peers in the same region (i.e. same sector) using P2P propagation.
   - Each server propagates these updates to the lower level hosts using TDP Propagation.

   In this technique, each peer has the responsibility of the resolution of update conflicts before propagating these updates to the other peers.

   As an example, in the zone area, this mechanism is applied as follows.
   i. The hosts in each cell propagate their updates to the cell server using BUP propagation.
   ii. The cell servers exchange those updates using P2P propagation.
   iii. Each cell server propagates these updates to its underlying hosts using TDP propagation.

   However, updates are propagated to the zone server only when they should be propagated to another zone. This case implies exchanging of these updates between the zone servers and their underlying hosts using this mechanism and eliminating the role of the master server.

5.3 Performance Evaluation

The two proposed propagation techniques, which are Hierarchical-Concentrate and P2P-Concentrate are evaluated and compared with Roam propagation technique with regard to achieving propagation delay reduction. The required equations that characterize the updates propagation are developed analytically in this section for computing the update propagation delay. In the analysis, a single update operation is tracked. The description of the update propagation delay as a performance metric and the required equations for analyzing it as well as the evaluation are as follows.

Update Propagation Delay (UPD). It is measured based on the number of hops that are required for propagating an update from a replica to another replica. This is because measuring the exact time that is consumed in updates propagation depends on many complicated factors such as connectivity (bandwidth and network delays) and availability of hosts. Moreover, mobile environments suffer from the inherited frequent disconnections, accordingly, we cannot rely on the actual propagation times and delays from a host to another.

Definition 2. UPD is the total number of hops from the host that represents the source of update to another host that is either in the same area or in different area.

Measuring UPD. To measure the propagation delay, we analytically developed the required equations that are based on the following assumptions:

(i) Two replicas: a replica on MHi, which generates an update that must be propagated to all other hosts. The other replica is MHj, which acts on behalf of all other hosts in that the same results are applied as they have been examined.

(ii) Two cases for the location of the destination, which are as follows.
   - Worst case: The purpose of this case is to determine the maximum number of hops that are required to propagate an update to all hosts. Therefore, the location of MHj (i.e. the destination) is assumed in the last cell, which exists in the last zone, or last master area, or it represents the last mobile host in the same cell of the MHi.
   - Average case: In this case, UPD is calculated on average regardless of the location of the MHj.

The required equations are developed by considering both worst and average cases for each propagation technique in a separate manner as follows.

(a) Measuring UPD for Hierarchical-Concentrate propagation

In Hierarchical-Concentrate propagation, the following equation is applied for both worst case and average cases.

$$\text{UPD} = \begin{cases} 1, \text{m} = 0, \text{z} = 0, \text{c} = 1 \\ 3, \text{m} = 0, \text{z} = 1, \text{c} \geq 2 \\ 5, \text{m} = 1, \text{z} \geq 2 \\ m + 4, m \geq 2 \end{cases} \quad (1)$$

Where:
- m is the number of master servers.
- z is the number of zone servers.
- c is the number of cell servers.

Proof. As provided at the appendix

Same values of UPD are hold in despite of the number of the cell where the update occurs (MHi exists) or the number of the other cell where MHj exists. This means that values do not change for different number of cells in the zone and different number of zones in the master area. This is in contrast with Roam.

(b) Measuring UPD for P2P-Concentrate propagation. In this technique, different equations are used for the worst case and the average case by considering that in each sector of the wheel rims, the peer can communicate only with the next nearby peer in the direction of the destination.

(i) Worst case

$$\text{UPD} = \begin{cases} n-2, \text{MHi and MHj in same cell}, \text{n} \geq 2 \\ c, \text{MHi and MHj in different cells in same zone}, \text{c} \geq 2 \\ z+2, \text{MHi and MHj in different zones in same master area}, \text{z} \geq 2 \\ m+4, \text{MHi and MHj in different master areas}, \text{m} \geq 2 \end{cases} \quad (2)$$

Where:
- n is the number of MHs in the cell.
- c is the number of CSs in the zone.
- z is the number of ZSs in the master area.
- m is the number of MSs.

Proof. As provided at the appendix

(ii) Average case

$$\text{UPD} = \begin{cases} \frac{1}{2}(n-2), \text{MHi and MHj in same cell}, \text{n} \geq 2 \\ \frac{c}{2}, \text{MHi and MHj in different cells in same zone}, \text{c} \geq 2 \\ \frac{1}{2}(z+2), \text{MHi and MHj in different zones in same master area}, \text{z} \geq 2 \\ \frac{1}{2}(m+4), \text{MHi and MHj in different master areas}, \text{m} \geq 2 \end{cases} \quad (3)$$

(c) Measuring UPD for Roam's propagation. In Roam, propagating an update from a replica MH_i to MH_j requires first sending it from MH_i to MH_j's ward master, then sending it from MH_i's ward master to MH_j's ward master, and then finally to MH_j. Accordingly, UPD is calculated as follows.

(i) Worst case

$$\text{UPD} = \begin{cases} n-2, \text{MHi and MHj in same ward} \\ w, \text{MHi and MHj in different wards} \end{cases} \quad (4)$$

(ii) Average case

$$\text{UPD} = \begin{cases} \frac{1}{2}(n-2), \text{MHi and MHj in same ward} \\ \frac{w}{2}, \text{MHi and MHj in different wards} \end{cases} \quad (5)$$

Where:
- n is the number of mobile hosts in the ward
- w is the number of wards

5.3.1 Comparative Study using ANOVA and Duncan's Test based on UPD

In this section, a comparative study for the three updates propagation techniques (i.e. Hierarchical-Concentrate, P2P-Concentrate, and Roam) is performed based on UPD as a performance metric. The purpose of the study is to determine the best technique among the three that can be used to propagate updates in large scale mobile distributed database systems.

The techniques are compared by varying the number of cells (i.e. equivalent to wards in Roam) and computing UPD based on the developed equations.

We assume that the number of cells in each zone is 5 and similarly, the number of zones in each master area is 5 (same conclusions are drawn when the number of cells or zones is greater than or equal 5 as already examined for different values). This means that in this comparison, varying the number of cells leads to the variation of the number of both zones and master servers in our strategy.

In this comparison, if MHi and MHj in the same cell, we consider that there are no any MHs between them. This because the number of MHs that act as hops between them cannot be estimated, since this depends on the number of MHs roaming at that cell on that time instant. Therefore, we consider UPD = 0 in this case as the best case for Roam.

Two replications for each cell number are taken into consideration for the calculation of UPD using the different techniques.

Accordingly, for this comparison, the following factors are considered:
1. Different techniques for updates propagation (Factor A)
2. Number of cells (Factor B)

A summary of the factors and their levels in the experimentation is presented in Table 1.

Table 1. Levels of the two factors A and B

Serial No.	Factors	Values	Number of Levels
1	Propagation techniques	---	3
2	Number of cells	1,2,3,…,100	100

Based on these factors, the experimental combination contains the number of the cell and the corresponding UPD values for the three techniques. Since two replications for each cell number (factor B) are taken into consideration for the calculation of UPD using the different levels of techniques (factor A), this means that the total number of experimental combinations is equal to 200.

The UPD values are analyzed in two stages using ANOVA and Duncan's multiple range tests [4]. The summary of these analyses is as follows.

Stage 1. ANOVA
The problem (i.e. comparing three techniques) is treated as two ANOVA.

- **Factor A:** Techniques
  Levels: 3
- **Factor B:** Number of cells
  Levels: 100
- **Response Variable:** Performance metric (measure or value) namely UPD.

- **Number of observations (n):** 600 (3*100*2)
- **Model:**
  The model of 2-factor experiment is as follows.

$$Y_{ijk} = \mu + \alpha_i + \beta j + \alpha\beta_{ij} + \varepsilon_{ijk} \quad (i=1, 2, 3; j=1, 2, 3,\ldots, 100; k=1,2) \qquad (6)$$

  Where:
  - Y_{ijk} is the performance measure namely UPD of the k^{th} replicate under the i^{th} and j^{th} treatments of the factors A and B respectively
  - μ is the overall mean effect.
  - α_i is the effect of the performance measure namely UPD due to the i^{th} treatment of Factor A.
  - βj is the effect of the performance measure namely UPD due to the j^{th} treatment of Factor B.
  - $\alpha\beta ij$ is the effect of the performance measure namely UPD due to the i^{th} treatment of Factor A and j^{th} treatment of Factor B.
  - ε_{ijk} is the random error (the effect of random experimental error)
- **Null hypotheses:**
  $\boldsymbol{H_0^1}$**:** $\alpha_1 = \alpha_2 = \alpha_3 = 0$
  Three techniques (Factor A) do not have significant effect on UPD.
  $\boldsymbol{H_0^{2:}}$ $\beta_1 = \beta_2 = \ldots = \beta_{100} = 0$
  Number of cells (Factor B) does not have significant effect on UPD.
  $\boldsymbol{H_0^3}$**:** $(\alpha\beta)_{ij} = 0$ for all i,j
  Interaction between techniques (Factor A) and number of cells (Factor B) does not have significant effect on UPD.
- **Alternative hypotheses:**
  $\boldsymbol{H_1^1}$**:** at least one $\alpha_i \neq 0$
  $\boldsymbol{H_1^2}$**:** at least one $\beta_i \neq 0$
  $\boldsymbol{H_1^3}$**:** at least one $(\alpha\beta)_{ij} \neq 0$
- **Level of Significance:** It is assumed as 0.05.
- **ANOVA Table:** It is as shown in Table 2. From this table, if the calculated value of F of a particular source of variation is greater than the corresponding tabulated value of F (F_T), then it can be concluded that the above source of variation is having significant effect on the performance measure namely UPD (i.e. the null hypothesis corresponding to the source of variation is rejected). Otherwise, it can be concluded that the source of variation is not having any significant effect on the performance measure namely UPD (i.e. the null hypothesis corresponding to the source of variation is accepted).
- **Results:** From the ANOVA statistics shown in Table 2, the following conclusions can be arrived at:
  a. Techniques (Factor A) are having significant effect on the performance measure namely UPD (i.e. $\boldsymbol{H_0^1}$ is rejected).
  b. Number of cells (Factor B) is having significant effect on the performance measure namely UPD (i.e. $\boldsymbol{H_0^2}$ is rejected).
  c. Interaction between Factor A and Factor B is having significant effect on the performance measure namely UPD (i.e. $\boldsymbol{H_0^3}$ is rejected).

  In accordance with ANOVA results, the model components A, B, and AB are statistically significant.

Table 2. Two way ANOVA table

Source of Variation	Sum Of Squares (SS)	Degrees Of Freedom (v)	Mean Sum Of Squares (MS)	F (Calculated)	F_T	$F > F_T$ Yes or No
A	139479.29	2	69739.65	496.1322	3.025846	YES
B	37930.74	99	383.14	2.725673	1.296908	YES
AB	56554.04	198	285.63	2.031964	1.234578	YES
E	42170	300	140.57			
Total	276134.1	599				

Stage 2. Test of means using Duncan's multiple range test. The first stage of the analysis concludes that the factor "Techniques" (Factor A) is having significant effect on UPD resulted into rejecting the null hypothesis. Accordingly, the next stage is the test of means to check whether the difference between any pair of treatment means is significant at a given confidence level. This stage is performed using Duncan's multiple range test.

Now, the steps that are carried out for this test are as follows.

1. Arranging the means in the ascending order of their respective values as shown in Table 3.
2. Calculation of the standard error of each average:

$$S = (^{MSE}/n)^{1/2} \quad (7)$$

Where:
- *MSE* is mean sum of square error from ANOVA (i.e. *MSE*= 210.85)
- *n* is the sample size (i.e. n= 200)

Thus S= $(210.85/200)^{1/2}$ = 1.027

3. Finding the critical value $q_\alpha(k, v)$ from the table of significant ranges [4]

Where:
- α is the significance level
- *k* the number of means being compared, and all means in-between (k=2,3)
- *v* is the degrees of freedom for error from the ANOVA table.

Accordingly, the critical values are: $q_{0.05}(2,200) = 3.687$ and $q_{0.05}(3,200) = 3.843$

4. Calculating the value of the least significant range (R_k):

$$R_k = q_\alpha(k, v) * S \quad (8)$$

Accordingly, the least significant ranges are:
$R_2 = q_{0.05}(2,200) * S = 3.787$
$R_3 = q_{0.05}(3,200) * S = 3.947$

5. Calculating the actual differences between the different pairs of means and comparing them with the corresponding least significant ranges. the actual differences between the different pairs of means are:
$A_3 - A_2 = 33.03 > R_3$
$A_3 - A_1 = 31.61 > R_2$
$A_1 - A_2 = 1.42 < R_2$

Table 3. Ordering of mean value*s*

Technique	Mean Symbol	Mean Value	Order
Hierarchical-Concentrate	A1	6.38	2
P2P- Concentrate	A2	4.96	1
Roam	A3	37.99	3

Duncan's test results. According to the previous analysis it can be concluded that there are significant differences between two pairs of means. The remaining pair is not significantly different. Accordingly, one can come to the following conclusions:

- The Roam propagation technique (corresponding to the mean value: A_3 = 37.99) performs most badly than the other two techniques, which are Hierarchical-Concentrate (corresponding to the mean value: A_1 = 6.38) and P2P-Concentrate (corresponding to the mean value: A_2 = 4.96).
- There is no significant difference between Hierarchical-Concentrate and P2P-Concentrate techniques.
- P2P-Concentrate technique (corresponding to the mean value: A_2 = 4.96) performs better than Hierarchical-Concentrate (corresponding to the mean value: A_1 = 6.38), but this in the case that the ordering process can be performed at any peer and then delegated from a peer to another peer.

Since there is no significant difference between Hierarchical-Concentrate and P2P-Concentrate, we conclude that the Hierarchical-Concentrate technique can be used mainly for propagating updates within the same master area in order to perform the ordering process in a hierarchical manner. This achieves fair

conflict resolution for all updates that are generated on the lower levels by delegating the responsibility of the resolution to the server in the higher level, while we use the P2P-Concentrate technique for propagating updates between the master areas, since there is no higher level than the master area. In this case, the ordering process is performed in a peer-to-peer manner in that each peer orders the merged updates (i.e. the received updates in addition to its generated updates), and then delegates the responsibility of ordering to the other peers.

5.3.2. Comparison with N-ary Tree Based Updates Propagation Protocol

The purpose of this comparison is to determine which mode of extension (that characterizes the propagation protocol) is suitable for LMDDBSs in that it achieves lower values of UPD. The extension here refers to the method that the propagation protocol follows for accommodating new replicas and servicing them through providing them with recent updates in the case of the replication system scales up in terms of its size and the number of replicas. Two modes of extensions are compared, which are: vertical extension that characterizes N-ary Tree Based Propagation Protocol (NTPP) and horizontal extension that characterizes Wheel-Based Propagation Protocol (WPP). The vertical extension requires adding new replicas to the last level of the *N*-ary tree or placing them in new levels in the case of the last level is full (i.e. there are no empty places for accommodating new replicas). The horizontal extension implies creating new propagation wheels when the replication system covers new distant areas. The vertical extension also characterizes the other tree based protocols. However, here we concern on comparing with NTPP because it is the most recent. Moreover, same conclusions are applied for the other tree based protocols.

In this comparison, the worst case is assumed for both protocols. For NTPP, this case implies that MHj (the host that receives updates occurred in MHi) exists in the last level, while MHi (where update occurred) exists in the root of the tree as this protocol implies. For WPP, the worst case implies that both MHi and MHj exist in the last level of the propagation wheel (most outer rim) and the location of MHj is assumed in the last cell. Hierarchical-Concentrate propagation technique is considered for propagation updates between the different propagation wheels, since it is equivalent to P2P-Concenterate in the case of the number of propagation wheels is greater than 1. We assume the propagation technique that is used in NTPP is Top-Down propagation according to its characteristics.

Based on the worst case, the number of hops that is imposed by NTPP for propagating an update from the root of the tree to a host in the last level is L -2, where L is the number of levels of *N*-ary tree. In contrast, in WPP, the number of hops that are required for propagating an update from MHi to MHj in another cell is fixed (as proved above), which equals 5 in the case of the number of master servers is 1 and the number of zones is greater than or equals 2.

For the sake of equivalent conditions for both NTPP and WPP, we assume that the number of propagation wheels increases as the levels of *N*-ary tree increase. This means that each propagation wheel is equivalent to a certain number of levels of *N*-ary tree. Accordingly, the number of propagation wheels is varied by considering the following cases:

(i) 1 propagation wheel is equivalent to 4 levels of *N*-ary tree.
(ii) 1 propagation wheel is equivalent to 5 levels of *N*-ary tree.
(iii) 1 propagation wheel is equivalent to 6 levels of *N*-ary tree.
(iv) 1 propagation wheel is equivalent to 7 levels of *N*-ary tree.

This variation is based on the aforementioned characteristic of the WPP that it extends in a horizontal manner as the replication system covers new areas, while NTPP extends in a vertical manner to encompass new hosts. By considering this variation, UPD values for each case of the above four cases are calculated based on the following equations:

- For NTPP:

$$UDP = L-2 \quad (9)$$

Where L is the number of levels

- For WPP:

$$UDP = 4+m \quad (10)$$

Where m is the number of propagation wheels as in equation 1 and 2 for measuring UPD for both Hierarchical-Concentrate and P2P-Concentrate (Note that one propagation wheel means one master server).

Based on the calculated values of UPD for both protocols, the following conclusions can be reached which are shown in Figure 3.

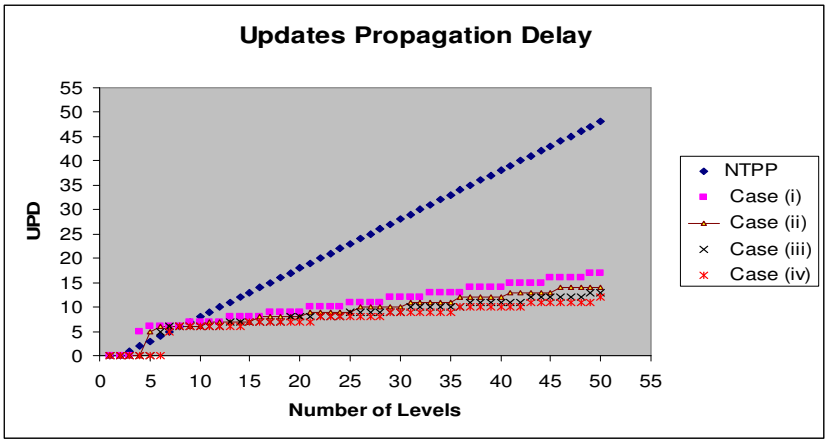

Figure 3. Comparison of the NTPP and the four cases of WPP based on the Updates Propagation Delay for the worst case

In Figure 3, NTPP has higher values for UPD than WPP for any case of the four cases when the number of levels gets higher. The value of this metric is slightly higher in case (i) than the other three cases when the number of levels gets higher. Also, it is slightly higher in case (ii) than the other two cases (i.e. case (iii) and case (iv)), and it is slightly higher in case (iii) than case (iv). This means that when the number of levels that are represented by one propagation wheel increases, the values of UPD get lower. Accordingly, the horizontal extension of WPP affects the value of UPD with a small amount of increasing (i.e. equals to 1) when a new wheel is constructed, while the vertical extension of NTPP increases the value of UPD with amount that is proportional to the number of new levels. Thus, the horizontal exertion is more suitable for LMDDBSs, since these systems extend by covering new areas. Moreover, placing new hosts of those areas in the last level of the *N*-ary tree is not reasonable due to the large values of UPD that are resulted by sending updates to these new hosts.

6. REPLICATION METHOD

The replication method is based on a new type of software agent called Instance-Immigration-Removed Agent (IIRA) that is introduced in this research. The research chose this name according to IIRA working nature, since it creates an instance of itself, and this instance migrates to another host and performs its task before removing itself. The purpose of the instance is to propagate recent updates that occurred or are collected in one level to another level in the replication architecture.

The following definition formally defines IIRA.
Definition 3. IIRA is a 5-tuple $< I, D, T, S, U >$, where:
$I = \{i_1, \ldots, i_n\}$ is a finite set of instructions that are required to perform IIRA tasks.
$D = \{d_1, d_2, d_3, d_4\}$ is a finite set of data structures that are involved in propagating recent updates between the components of the replication architecture, and collecting recent updates from underlying levels.
$T = \{t_1, t_2, t_3, t_4\}$ is a finite set of IIRA types. A type t_i maps each IIRA to a certain level (i.e. the type determines the location of IIRA).
$S = \{s_1, s_2, s_3, s_4, s_5, s_6\}$ is a finite set of IIRA states. Each state represents the current activity that is carried out by IIRA.
$U: T \rightarrow \{1, 2, 3, \ldots, k\}$ is a function for assigning a unique identifier for IIRA in the system.

According to this definition, the IIRA consists of code and database and it has type, state, and unique identifier.

IIRA Types. The paper divides IIRA into four types according to the level in which the IIRA carries out its activities. The four types share common structure and behavior, but they inhabit different levels. These types are as follows.

- **MH-Resident IIRA (MHR-IIRA).** Every MH has IIRA, and it is responsible for propagating recent updates that are performed on MH to the other hosts in the mobile network or to the cell server that covers the cell where the MH is located at the time of connection.
- **Cell Server-Resident IIRA (CSR-IIRA)**. This type acts as a broker for propagating recent updates between the fixed network and the mobile network. It propagates all updates that are received from MHR-IIRA to the zone server and vice versa. It is responsible for providing a unified ordering for all updates that occurred in its cell and maintaining total ordering of messages that are received from the higher level.
- **Zone Server-Resident IIRA (ZSR-IIRA).** This type receives all updates that are performed in the fixed network and the mobile network, which are associated with specific zone level, and provides a unified ordering for these updates on this level. Then, it propagates these updates directly to the master server. Also, this type provides underlying cell servers with a set of totally ordered updates that is received from the master server.
- **Master Server-Resident IIRA (MSR-IIRA)**. This type receives all updates that are performed in the zone level and provides a total unified ordering for all updates occurred in the replication system. Then, it propagates these ordered updates directly to each zone server, which in turn propagates these updates to underlying levels.

IIRA states. The possible states for IIRA are monitoring the connection with the other hosts, retrieving the set of recent updates, creating the instance, migration of the instance to the host that the connection is realized with it, execution of the instance on the other host, and removing the instance after completion of its task.

6.1. IIRA-based Propagation System

The purpose of this system is to implement the proposed propagation protocol through automating the propagation of updates between the components of the replication architecture in a manner that ensures bounding unavailability of recent updates and inconsistency of replicated data.

The IIRA-Based propagation system (see Figure 4) is composed of the four types of IIRA. Each type interacts with the other types through a synchronization process in which two types exchange their recent updates via their created instances of IIRA. In this system, each type can exchange updates directly with the same type or a different type in the server of the level where it inhabits or in a server from underlying level. For example, MHR-IIRA can exchange updates directly with CSR-IIRA or another MHR-IIRA, while MHR-IIRA cannot directly interact with either ZSR-IIRA or MSR-IRRA.

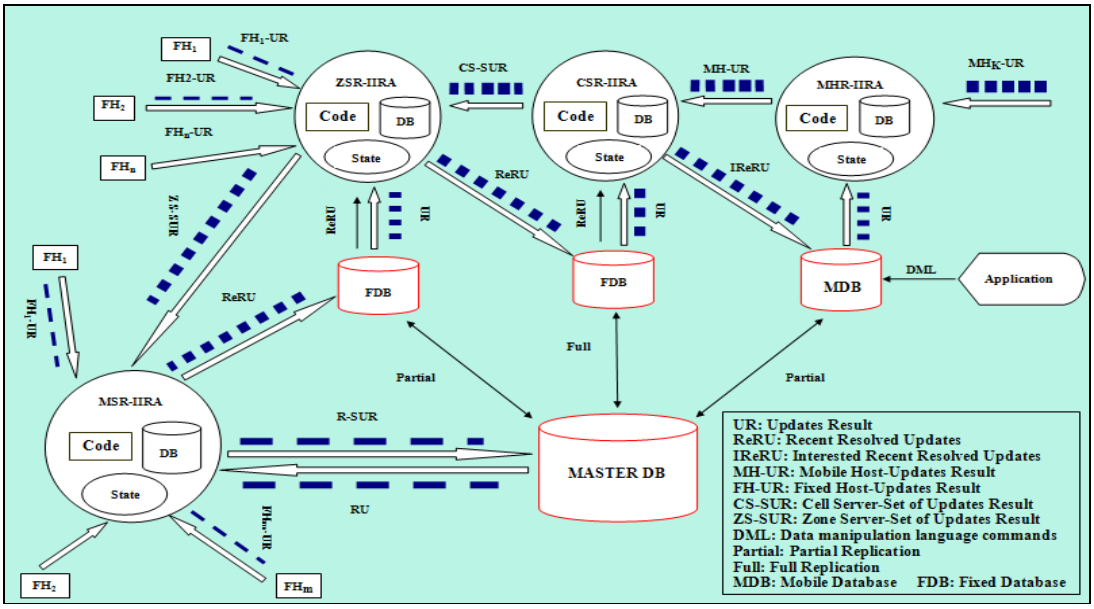

Figure 4. IIRA-Based propagation system

When the connection takes place between any two IIRA types, an instance of IIRA type that inhabits the lower level will propagate a set of recent updates called Updates-Result (UR), which is produced by an application, in addition to updates that may have been collected from the underlying level to the database of IIRA type that inhabits the higher level. Then, an instance of the type that inhabits the higher level propagates a set of updates called Recent-Resolved-Updates (ReRU) that is received from the higher level to the replicated database in the lower level. Accordingly, IIRA has different responsibilities for updates propagation with respect to implementing the three basic mechanisms of updates propagation (Bottom-Up Propagation, Top-Down Propagation, and Peer-to-Peer Propagation). These responsibilities are identified in the following subsection.

6.1.1. IIRA Responsibilities for Updates Propagation

The responsibilities of IIRA can be identified according to the mechanism of the propagation as follows.
Bottom-Up Propagation: In this type, each MHR-IIRA propagates the set of recent updates (MH-UR) that occurred in its host to the database of the type that inhabits a cell server. Also, each server's IIRA collects the received recent updates from underlying level in addition to its server's updates-result in a set called Set of Updates Result (SUR), and resolves updates conflict through ordering updates in this set, and then propagates this set to the database of IIRA in the higher level.

The typical operations that are involved in this propagation, which are performed by IIRA are:

- Resolving updates conflict through ordering the collected updates, which are stored on the database of the IIRA type that inhabits the host in the current level.
- Creating the instance and filling it with the ordered updates.
- Migration of the instance to the host that exists in the higher level.
- Execution of the instance in the destination host through insertion of recent updates in its IIRA type's database.

Thus, this propagation takes place in the servers that exist in the fixed network when IIRA orders the collected updates. Then, the ordered updates are shipped via the created instance to the host in the higher level. This means that updates are pushed from a level to the higher level in a unicasting manner where each host pushes them to a single host in the higher level. Updates occurred on mobile hosts are pushed to cell servers when the connection occurs.

Top-Down Propagation: In this type, each server's IIRA propagates the set of recent resolved updates (ReRU) that is received from the higher level to the lower level. For example, each ZSR-IIRA propagates the ReRU that is received from the master server to underlying cell servers and in turn each cell server propagates a subset of this set called Interested-Recent Resolved-Updates (IReRU) to underlying mobile hosts. IReRU represents the set of updates that are performed on the data items that are replicated on mobile hosts.

The typical operations that are performed by IIRA for carrying out this propagation are:

- Creating the instance and storing the set of recent resolved updates on it.
- Migration of the instance to the host that exists in the lower level.
- Execution of the instance on the destination host.

Thus, for a given set of updates, this propagation takes place when these updates are totally ordered (i.e. resolved) by MSR-IIRA. The ordered updates (i.e. ReRU) are then propagated through the instance to the underlying level. The MS pushes ReRU to all available zone servers through sending the same instance to each zone server. Similarly, the zone server does same pushing to its underlying available cell servers. However, the cell servers push ReRU only to connected MHs, while disconnected MHs pull ReRU when they connect to the cell servers. According to this pushing mechanism, this mechanism of propagation can be called Multi-cast Propagation.

Peer-to-Peer Propagation: In this type, two IIRA of the same type may exchange their updates. The typical operations that are performed by IIRA in this type are same as in Bottom-Up propagation with a distinction that both hosts, which are involved in the synchronization process inhabit the same level and have same type.

7. CONCLUSIONS

This paper presented a data replication strategy for maintaining the eventual consistency of replicated data in large scale mobile distributed database systems with large number of updates. The strategy encompassed three-level replication architecture, wheel-based updates propagation protocol, and the replication method as a ternary combination that is needed to act in accordance with the characteristics of such systems. The strategy supports frequent disconnections and mobility of hosts by enabling the users to perform their updates in a disconnected mode and then synchronizing their updates with the higher levels.

The effectiveness of the proposed strategy with respect to updates propagation is verified through the comparative study with Roam replication system, which represents one of the fewer replication systems that are devoted to large scale mobile environments. The results revealed that the proposed strategy achieves better propagation delay than Roam replication system. The better propagation delay represents an essential feature that characterizes scalable replication strategies with regard to updates propagation to all replicas.

Also, the proposed updates propagation protocol is compared with *N*-ary tree based propagation protocol, which represents one of the most recent tree based propagation protocols. The results showed that the horizontal extension provided by the proposed protocol is more suitable than the vertical extension for LMDDBSs, since such type of systems is extended to cover new geographic areas. Moreover, placing new hosts of those areas in a tree structure with a variable number of levels as in *N*-ary tree based propagation protocol is not reasonable due to the large values of updates propagation delay that are resulted by sending updates to the new hosts, especially when they are placed in either the last level of the tree or a new established level.

As a part of our future research, a plan will be provided to develop the required specifications, tools, and interfaces to implement the proposed strategy in large-scale mobile healthcare environments to provide healthcare practitioners with an efficient access to updates that are performed on healthcare data.

REFERENCES

[1] A. Helal, A. Heddaya and B. Bhargava, (1996) "Replication techniques in distributed systems", Kluwer Academic Publishers.

[2] A. Waluyo, B. Srinivasan and D. Taniar, (2005) "Research in mobile database query optimization and processing", *Mobile Information Systems*, 1(4), 225–252.

[3] B. Nishio and M. Tsukamoto, (2002) "Data management issues in mobile and peer-to-peer environments", *Data & Knowledge Engineering*, 41(2–3), 183–204.

[4] D.C. Montgomery, (1991) "Design and analysis of experiments", 3rd edition, John wiley & Sons.

[5] D. Barbara, (1999) "Mobile computing and databases – a survey", *IEEE Transactions on Knowledge and Database Engineering*, 11(1), 108-117.

[6] D. Ratner, P. Reiher, G. Popek and G. Kuenning, (2001) "Replication requirements in mobile environments", *Mobile Networks and Applications* 6(6), 525–533.

[7] D. Ratner, (1998) "Roam: a scalable replication system for mobile and distributed computing", PhD Dissertation, University of California, Los Anglos.

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

[9] D. Zhiming, M. Xiaofeng and W. Shan, (2002) "A transactional asynchronous replication scheme for mobile database systems", *Journal of Computer Science and Technology*, 17(4), 389 – 396.

[10] G. Sushant, and R. Buyya, (2006). "Data replication strategies in wide-area distributed systems", Chapter IX of Enterprise Service Computing: From Concept to Deployment, IGI Global, pp. 211-241.

[11] H. Yu and A. Vahdat, (2000) "Design and evaluation of a continuous consistency model for replicated services", In proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI), San Diego, CA, USA, pp. 305–318.

[12] J.Abawajy, M. Deris and M. Omer, (2006) "A novel data replication and management protocol for mobile computing systems", *Mobile Information Systems*, 2(1), pp. 3-19.

[13] J. Holliday, D. Agrawal and A. Abbadi, (2002) "Disconnection modes for mobile databases", *Wireless Networks* 8(4), pp. 391-402.

[14] J. Monteiro, A. Brayner and S. Lifschitz, (2007) "A Mechanism for replicated data consistency in mobile computing environments", In Proceedings of the ACM symposium on Applied computing, Seoul, Korea, 914 – 919.

[15] J.P. Barreto, (2003) "Information sharing in mobile networks: a survey on replication strategies", Technical Report RT/015/03, Instituto Superior T´ecnico, Lisboa.

[16] M. Stonebraker, (1979) "Concurrency control and consistency of multiple copies of data in distributed INGRES", *IEEE Transactions on Software Engineering*, SE-5, pp. 188–194.

[17] N. Tolia, M. Satyanarayanan and A. Wolbach, (2007) "Improving mobile database access over wide-area networks without degrading consistency", In Proceedings of the 5th international conference on Mobile systems, applications and services, San Juan, Puerto Rico, pp. 71 – 84.

[18] P.A. Bernstein, V. Hadzilacos and N. Goodman, (1987) "Concurrency control and recovery in database systems", Addison-Wesley.

[19] S. Madria and S. Bhowdrick, (2001) "Mobile data management", *Potentials, IEEE,* 20(4), pp. 11 – 15.

[20] S. Phatak and B. Nath, (2004) "Transaction-Centric reconciliation in disconnected Client-Server databases", *Journal of Mobile Networks and Applications*, 9(5), pp. 459–471.

[21] S.R. Terry, (1983) "Stronger than a hundred men: a history of the vertical water wheel", Johns Hopkins University Press.

[22] T. Connolly and C.E. Begg, (2004) "Database systems: a practical approach to design, implementation and management", 4th edition, Addison-Wesley.

[23] T. Ekenstam, C. Matheny, P. Reiher and G. Popek, (2001) "The bengal database replication system", *Distributed and Parallel Databases,* 9(3), 187-210.

[24] T. Imielinksi and B. Badrinath, (1994) "Wireless mobile computing: challenges in data management", *Communications of ACM*, 37(10), 18-28.

[25] T. Hara, M. Nakadori, W. Uchida, K. Maeda, and S. Nishio, (2005) "Update propagation based on tree structure in peer-to-peer networks", In proceedings of AICCSA, pp. 40–48.

[26] T. Ozsu and P. Valduriez, (1999) "Principles of distributed database systems", 2nd Edition, Prentice-Hall.

[27] T. Watanabe, T. Hara, Y. Kido and S. Nishio, (2007) "An update propagation strategy for delay reduction and node failure tolerance in peer-to-peer networks", In Proceedings of the 21st IEEE International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), pp. 103-108.

[28] T. Watanabe, K. Akimitsu, T. Hara and S. Nishio, (2008) "An update propagation strategy considering access frequency in peer-to-peer networks", DASFAA 2008, LNCS 4947, Springer-Verlag Berlin Heidelberg, pp. 661–669.

[29] Y. Breitbart and H. Korth, (1997) "Replication and consistency: being lazy helps sometimes", In Proceedings of 16 ACM Sigact/Sigmod Symposium on the Principles of Database Systems, Tucson, Arizona, 173 – 184.

[30] Y. Saito and M. Shapiro, (2005) "Optimistic replication", *ACM Computing Surveys*, 73(1), pp. 42 - 81.

Appendix

1. Proof for equation of Measuring UPD for Hierarchical-Concentrate

Assume that h is the number of hops. The following cases are considered:

a. If c=1 → h=1

b. If c=2 → h=3

This is because if c = 2, this implies existing of one zone server according to our assumption that two or more cell servers need a zone server for resolving their conflicts.

Thus, c = 2 ↔ z =1

c. If z =2 → h=5

Also, if z= 2, this implies existing of one master server for resolving their conflicts.

Thus, z = 2 ↔ m=1

d. For $m \geq 2$, we use mathematical induction as follows:

If m=2 → h=*6*

This is because updates should be propagated in P2P manner in case of existing more than one master server since there is no higher level than the master level in our strategy. Update conflicts are resolved by delegating the responsibility of resolving to the next peer.

Accordingly, the If m = k → h=*k+4*

Thus, if m = k +1 → h=*(k+1)+4*

2. Proof for equation of measuring UPD for P2P-CONCENTRATE

By using mathematical induction and assuming h is the number of hops, we consider the following cases:

a. MH_i and MH_j in the same cell

If n=2 → h=*0* (There are no hops between MH_i and MH_j)

If n=3 → h=*1*

Accordingly, If n=k → h=*k-2*

Since the equation holds for n=k, this implies that:

If n=k+1 → h= (k- 2) +1= (k+1)-2

b. MH_i and MH_j in different cells in the same zone

If c=2 → h=2

If c=k → h=k

Thus, if c= k+1 → h=k+1

c. MH_i and MH_j in different zones in the same master area

If z=2 → h=4

If z=k → h=k+*2*

Thus, if z= k+1 → h= (k+1) +2

d. MH_i and MH_j in different master areas

The proof is performed in same manner as above.