A Binary Replication Strategy for Large-scale Mobile Environments

Ashraf A Fadelelmoula ashrafafadel@hotmail.com
Department of Computer and Information Sciences
Universiti Teknologi PETRONAS
31750 Tronoh, Perak, Malaysia

P.D.D.Dominic dhanapal_d@petronas.com.my
Department of Computer and Information Sciences
Universiti Teknologi PETRONAS
31750 Tronoh, Perak, Malaysia

Azween Abdullah Azweenabdullah@petronas.com.my
Department of Computer and Information Sciences
Universiti Teknologi PETRONAS
31750 Tronoh, Perak, Malaysia

Hamidah Ibrahim hamidah@fsktm.upm.edu.my
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia

ABSTRACT

An important challenge to database researchers in mobile computing environments is to provide a data replication solution that maintains the consistency and improves the availability of replicated data. This paper addresses this problem for large scale mobile environments. Our solution represents a new binary hybrid replication strategy in terms of its components and approach. The new strategy encompasses two components: replication architecture to provide a solid infrastructure for improving data availability and a multi-agent based replication method to propagate recent updates between the components of the replication architecture in a manner that improves availability of last updates and achieves the consistency of data. The new strategy is a hybrid of both pessimistic and optimistic replication approaches in order to exploit the features of each. These features are supporting higher availability of recent updates and lower rate of inconsistencies as well as supporting the mobility of users. To model and analyze the stochastic behavior of the replicated system using our strategy, the research developed Stochastic Petri net (SPN) model. Then the Continuous Time Markov Chain (CTMC) is derived from the developed SPN and the Markov chain theory is used to obtain the steady state probabilities.

Keywords: pessimistic replication, optimistic replication, availability, consistency, Stochastic Petri net.

1. INTRODUCTION

Rapid advancements in wireless technologies and mobile devices have made mobile computing enjoying considerable attention in the past few years as a fertile area of work for researchers in the areas of database and data management. As mobile computing devices become more and more common, mobile databases are also becoming popular. Mobile database has been defined as 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 corporate data [1].

Mobility of users and portability of devices pose new problems in the management of data [2, 3], 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 [4]. 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 and improving availability of replicated data for large scale distributed database systems that operate in mobile environments. This type of systems is characterized by a large number of replicas (i.e. hundreds of replicas) and a large number of updates (i.e. tens of updates per data items are expected at any period of time) are performed in these replicas. Examples of such systems include mobile health care, mobile data warehousing, news gathering, and traffic control management systems.

In such type of mobile environments, the concurrent updates of large number of replicas during the disconnection time influences consistency and availability of the replicated data, by leading to divergence in the database states (i.e. the data in the database at a particular moment in time). As a solution to the aforementioned problems, this paper proposes a new replication strategy that acts in accordance with the characteristics of large scale mobile environments.

This paper is organized as follows. The next section provides the background and related work. Section 3 describes the proposed replication strategy. Section 4 gives the details of the behavior modeling. Section 5 presents the contribution and discussions. Section 6 concludes the paper.

2. BACKGROUND AND REALTED WORK

Data replication strategies are divided into optimistic and pessimistic approaches [5, 6, 7]. 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 failures uncommon. Primary-copy algorithms [8] 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.

Using optimistic replication in mobile environments has been studied in several research efforts. ROAM [9] is an optimistic replication system that provides a scalable replication solution for the mobile user. ROAM is based on the Ward Model [10]. The authors group replicas into wards

(wide area replication domains). All ward members are peers, allowing any pair of ward members to directly synchronize and communicate.

A multi-master scheme is used in [11], 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, an adaptation in the primary commit scheme is used.

A hybrid replication strategy is presented in [12] 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 object is replicated asynchronously at only one site based on the most frequently visited site.

Cedar [13] 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 occur hours or days apart), its replica is refreshed from the master copy.

However, aforementioned strategies have not explicitly addressed the issues of consistency and availability of data in large scale distributed information systems that operate in mobile environments. Therefore, this paper comes to a conclusion that additional research toward a new replication strategy is needed to investigate and address above mentioned issues.

2.1 SPN Background

Petri Nets (PNs) are an important graphical and mathematical tool used to study the behavior of many systems. They are very well-suited for describing and studying systems that are characterized as being concurrent, asynchronous, distributed, and stochastic [17, 19]. A PN is a directed bipartite graph that consists of two types of nodes called places (represented by circles) and transitions (represented by bars). Directed arcs connect places to transitions and transitions to places. Places may contain tokens (represented by dots).

The state of a PN is defined by the number of tokens contained in each place and is denoted by a vector M, whose i^{th} component represents the number of tokens in the i^{th} place. The PN state is usually called the PN marking. The definition of a PN requires the specification of the initial marking M'. A place is an input to a transition if an arc exists from the place to the transition. A place is an output from a transition if an arc exists from the transition to the place. A transition is said to be enabled at a marking M when all of its input places contain at least one token. A transition may fire if it is enabled. The firing of a transition t at marking M removes one token from each input place and placing one token in each output place. Each firing of a transition modifies the distribution of tokens on places and thus produces a new marking for the PN.

In a PN with a given initial marking M', the reachability set (RS) is defined as the set of all markings that can be "reached" from M' by means of a sequence of transition firings. The RS does not contain information about the transition sequences fired to reach each marking. This information is contained in the reachability graph, where each node represents a reachable state, and there is an arc from M_1 to M_2 if the marking M_2 *is* directly reachable from M_1. If the firing of t led to changing M_1 to M_2, the arc is labeled with t. Note that more than one arc can connect two nodes (it is indeed possible for two transitions to be enabled in the same marking and to produce the same state change), so that the reachability graph is actually a multigraph.

SPNs are derived from standard Petri nets by associating with each transition in a PN an exponentially distributed firing time [16, 18]. These nets are isomorphic to continuous-time Markov chains (CTMCs) due to the memoryless property of exponential distributions. This property allows for the analysis of SPNs and the derivation of useful performance measures. The states of the CTMC are the markings of the reachability graph, and the state transition rates are

the exponential firing rates of the transitions in the SPN. The steady-state solution of the equivalent finite CTMC can be obtained by solving a set of algebraic equations.

3. REPLICATION STRATEGY

The proposed replication strategy encompasses two components: replication architecture and replication method. The purpose of the replication architecture is to provide a comprehensive infrastructure for 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 replication method is to transfer data updates between the components of the replication architecture in a manner that achieves the consistency of data and improves availability of recent updates to interested hosts.

The new strategy is a hybrid of both pessimistic and optimistic replication approaches. The pessimistic approach is used for restricting updates of infrequently changed data to a single replica. The reason behind this restriction is that if the modifications of these data are allowed on several sites, it will influence data consistency by having multiple values for the same data item (such as multiple codes for the same disease or multiple codes for the same drug). On the other hand, the optimistic replication is used for allowing updates of frequently changed data to be performed in multiple replicas. The classification into frequently and infrequently changed data is specified according to the semantic and usage of the data items during the design phase of the database.

3.1. System Model

This research considers a large-scale environment consists of fixed hosts, mobile hosts, a replica manager on each host, and a replicated database on each host. A replicated database is called mobile database when it is stored in a mobile host. A part of fixed hosts represent servers with more storage and processing capabilities than the rest. 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.

Definition 3.1.1 An object O is the smallest unit of replication and it represents a tuple $O = <D, R, S>$, where $D = \{d_1, d_2, \ldots, d_n\}$ is a set of data items of the object O, $R = \{r_1, r_2, \ldots, r_m\}$ is a set of replicas of O, and S is the state of the object O.

Definition 3.1.2 The state S of an object O is a set consisting of states that identifies current values for each data item $d_i \in D$, i.e., $S = \{s_1, s_2, \ldots, s_n\}$.

Definition 3.1.3 A replica R is a copy of an object stored in a different host and is defined as a function as follows. For a set of updates U that is performed on a set of objects $\bar{O}$, the function $R : U \times \bar{O} \rightarrow S$ identifies a new separate state $s_i \in S$ for an object $O \in \bar{O}$ as a result of performing update $u \in U$ on an object O in a different host.

Definition 3.1.4 For a replica R in a host H, Interested Data Items is a subset I of the set of all data items, which is required for H to perform its duties, i.e., $I \subseteq \{ \bigcup_{i=1}^{n} O_i \}$, where n is the number of objects in the system.

Definition 3.1.5 A replicated data item $d_i \in D$ is consistent if and only if its values are identical and in same order as the values of the similar data item in the replica that is stored in the master server, which exists in the fixed network

Definition 3.1.6 A replica R is consistent if and only if each interested data item $d_i \in \{D \cap I\}$ is consistent.

Definition 3.1.7 A replica R in a mobile host is in Available-State for a data item $d_i \in D$ if and only if all updates that are performed on d_i in other replicas (either in fixed hosts or mobile hosts) are merged with the updates that are performed on d_i in R.

Definition 3.1.8 Consistent-Available State (CA State) for a replica R that is stored in a mobile host is the state in which:

1. R is consistent
2. R in Available-State for each interested data item d_i in R.

3.2 Replication Architecture

The proposed replication architecture considers a total geographic area called the master area, divided into a set $Z = \{z_1, \ldots, z_n\}$ of zones. Each zone consists of a set $C = \{c_1, \ldots, c_m\}$ of smaller areas called cells (see figure 1). Each cell represents an area, where the mobile users can perform their duties at a particular period of time before moving to another cell. In this architecture, the network is divided into fixed network and mobile network. The fixed network consists of Fixed Hosts (FH) and wired local area network to connect the fixed hosts in the master area, and also include wide area network to connect fixed hosts in the master and zone areas, and the servers of the cell area. 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 (MH) in the cell area.

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

Master Level: This level contains the master replica, which must be synchronized with the replicas from the zone level. The server in this level is responsible for synchronizing all changes that have been performed on infrequently 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 data 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.

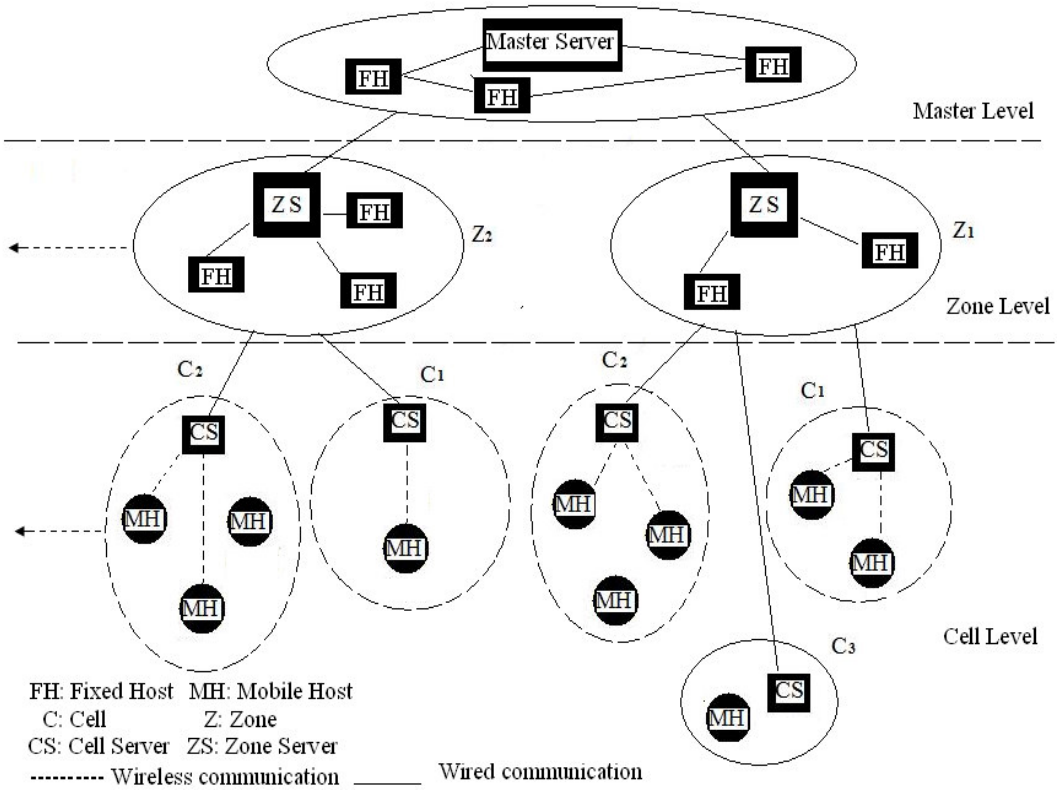

FIGURE 1: The Replication Architecture for Mobile Environments

3.3. Replication Method

The replication method is based on a multi-agent system called IIRA-dependant multi-agent system. This system is proposed 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 other level in the proposed replication architecture. The following definition will formally define IIRA.

Definition 3.3.1 IIRA is a 5-tuple $<T, S, D, I, U>$, where:

$T= \{t_1, t_2, t_3, t_4\}$ 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.

$D = \{d_1, \ldots, d_n\}$ is a finite set of data items that are required to store recent updates that are performed on the similar data items, which are stored in the database.

$I = \{i_1, \ldots, i_n\}$ is a finite set of primitives/instructions that are required to perform IIRA activities and the transitions.

$U: T \rightarrow \{1, 2, 3, \ldots, k\}$ is a function for assigning a unique identifier for IIRA in the system.

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

IIRA Types: The research 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:

MH-Resident IIRA (MHR-IIRA): Every MH has IIRA, and it is responsible for propagating recent updates that are performed in the MH to 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.

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 resolves update conflicts on this level. Then it propagates these updates directly to the master server.

Master Server-Resident IIRA (MSR-IIRA): This type receives all updates that are performed in the zone level and resolves update conflicts. Then it propagates these updates directly to each zone server, which in turn propagates these updates to underlying levels.

IIRA States: The possible states of the IIRA in the proposed replication strategy are:

Monitoring: In this state, the IIRA monitors the connection with the other devices through interacting with its environment (i.e. hosted device) via message passing.

Retrieving: The IIRA retrieves the set of recent updates from the hosted device. The IIRA enters this state when the monitoring state results in a connection that is realized with the other host.

Creating Instance: The IIRA creates an instance of it and stores the set of recent updates on this instance.

Migration: The IIRA instance migrates from the hosted device to other device that the connection is realized with it.

Insertion: In this state, the IIRA instance inserts its stored recent updates in the database of the IIRA in the other device.

Removing: The migrated instance of IIRA removes itself after completion of the insertion process.

3.3.1 IIRA-Dependant Multi-Agent System

The IIRA-dependent multi-agent system (see figure 2) is composed of the four types of IIRA. Each type interacts with others through a synchronization process, in which two types exchange their recent updates via their created instances. 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 (e.g. the MHR-IIRA can exchange updates directly with CSR-IIRA or other MHR-IIRA).

The exchanging of updates between two types occurs only during the connection period between their owner hosts. The time period that MH waits for the connection with a cell server in the fixed network is not deterministic and it depends on its own conditions, such as connection availability, battery, etc. On the other hand, the research assumes that the time period that a server in the fixed network must wait to connect with the server in the higher level is deterministic and its value depends on the availability and consistency requirements of the replicated system. To decide this value, the research views the connection timing system for connecting servers in the fixed network should mimics the shifting behavior, where the shift time is exploited in collecting recent updates from underlying level (e.g. the cell server should connect to the zone server every one hour to propagate the collected updates from underlying mobile hosts during the last past hour).

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. According to this fact, the research defines three types of 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 Updates-Result from underlying level in addition to its server's updates-result in a set called Set of Updates Result (SUR) and resolves updates conflict in this set and then propagates this set to the database of IIRA in the higher level.

As previously mentioned, the time period that the server in the current level should wait for receiving recent updates (i.e. updates collection) from underlying level is deterministic. The value of this period in a server in the higher level is greater than its value in a server in the lower level (e.g. waiting period for the zone server > waiting period for the cell server). This is because the number of hosts, where updates occurred increases as we move to the higher levels. After elapsing of this deterministic period, the IIRA carries out the required operations for this type of propagation.

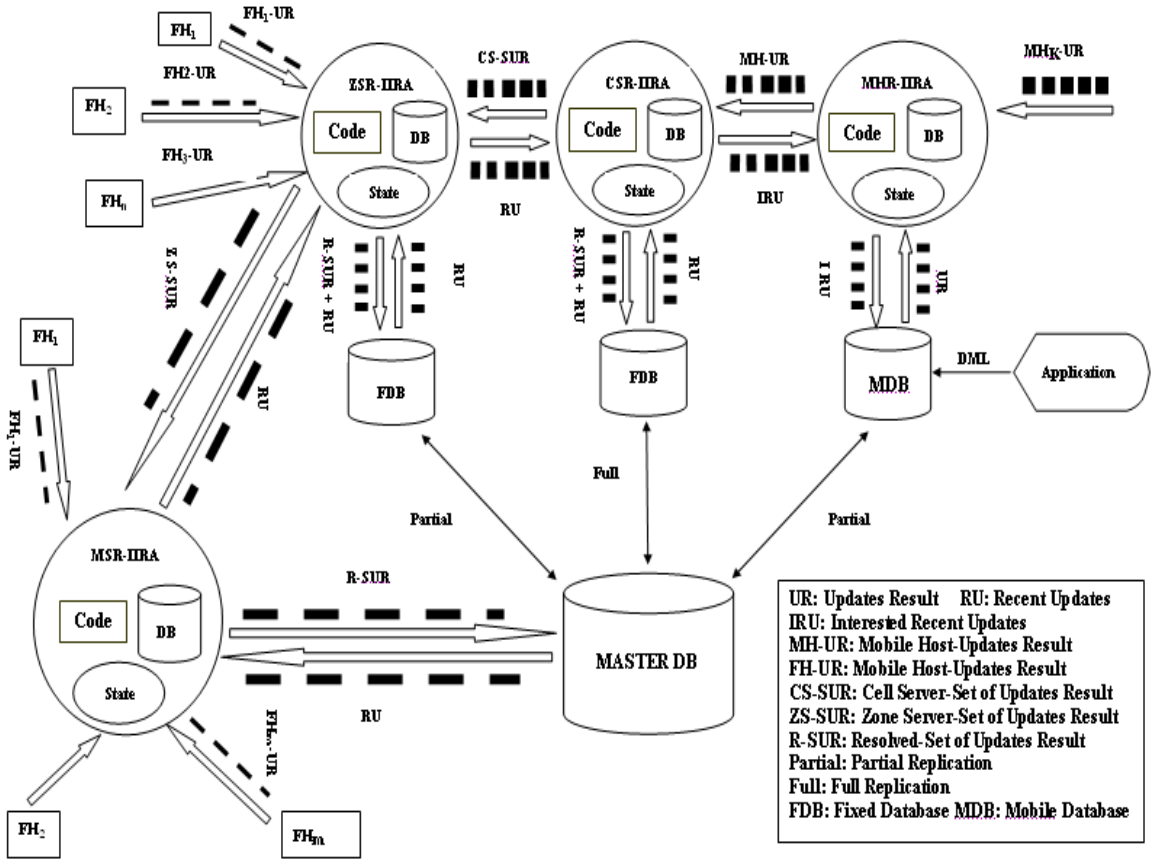

FIGURE 2: IIRA-Dependant Multi-Agent System

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

- Resolving updates conflict through ordering the collected updates on the database of the IIRA type that inhabits the host in the current level and assigning the value of the update timestamp data item (in case of the update is generated on the same host).
- Creating the instance and storing the ordered updates on it.
- Migration of the instance to the host that exists in the higher level and assigning the value of the send timestamp data item to each update before the migration start and after migration request is accepted.
- Execution of the instance in the destination host through insertion of recent updates in its IIRA type's database.

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 the 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.

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 in the destination host.

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, are of the same level.

4. BEHAVIOR MODELING

The paper models the dynamic behavior of the proposed multi-agent system with respect to the synchronization process between its components by using SPNs. The purposes are to trace how the mobile database will reach the CA state and to calculate the steady-state probabilities.

The reason of using SPNs is that the research views the synchronization process between the different levels of the replication architecture as a discrete-event stochastic system that encompasses states and events. Each state captures either a snapshot of a set of recent updates or a snapshot of a set of tuples currently is stored in the database. Each event represents either execution of IIRA instance in another level or retrieving of a subset of tuples. The system is stochastic due to its stochastic state transitions since it is evolving over continuous time and making state transitions when events associated with states occur.

The behavior modeling approach using SPN follows a systematic approach described in [14, 15], which incorporates the following steps.

- Modeling of the behavior of the IIRA-dependant multi-agent system using a stochastic Petri net.
- Transforming the developed SPN into its equivalent Markov chain for calculation of the steady state probabilities of marking occurrences. This step requires generating the reachability graph. The Markov chain is obtained by assigning each arc with the rate of the corresponding transition.
- Analyze the Markov chain to obtain steady state probabities.

The research interests in a state in which the mobile database in the mobile host receives a set of recent updates that occurred on the other hosts in both fixed and mobile networks.

4.1 The SynchSPN

The following definition will formally define the developed SPN that is used to model the behavior.
Definition 4.1. The SynchSPN is a six-tuple $<P, T, A, W, m_0, \Lambda>$ where:
1. $P = \{p_1, p_2, \ldots, p_{11}\}$ is a finite set of places, each place represents either a synchronization state for IIRA database (IIRA-Synch-State**)** or a synchronization state for the replicated database (DB-Synch-State**)** in each level. The former contains the set of recent updates, while the latter contains the set of tuples currently stored in the database.

2. $T= \{t_1, t_2, \ldots, t_{14}\}$ is a finite set of transitions, each transition represents an operation carried out by the IIRA in different levels. These operations are:
   i. Retrieving the set of recent updates.
   ii. Execution of the IIRA instance in the other level to transfer the recent updates.

3. $A \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs that represents the number of updates, which have to be transferred after the execution process or the number of updates that have to be retrieved after the retrieving process.

4. $W: A \rightarrow \{1, 2, \ldots\}$ is the weight function attached to the arcs. This function maps each arc to the number of updates that are to be propagated in each synchronization process between the different levels.

5. $m_0: P \rightarrow \{0, 0, 0, 0, |MH\text{-}DBS|i, |CS\text{-}DBS|j, |ZS\text{-}DBS|k, |MS\text{-}DBS|, 0, 0, 0\}$ is the initial marking, which represents the number of recent updates at the synchronization state for each IIRA (p_1, p_2,

p_3, p_4, p_9, p_{10}, p_{11}) and the number of tuples that are currently stored in each database (p_5, p_6, p_7 , p_8). (In the context of discrete-event systems, the marking of the PN corresponds to the state of the system).

6. $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_{14}\}$ is the set of firing rates associated with the SPN transitions.

SynchSPN is developed based on the following theorem.
Assertion 4.1. The synchronization process has a Markov property.

Proof. Let $U(t)$ denotes the number of recent updates that should be propagated by IIRA instance from a host to another during their synchronization at time instant t, where t varies over a parameter set T. The value of $U(t)$ is not deterministic because it depends on the number of generated updates, which means $U(t)$ is a random variable. Accordingly, the synchronization process can be defined as a family of random variables $\{U(t)|t \in T\}$, where the values of $U(t)$ represent the states of the synchronization process. Thus, the synchronization process represents a stochastic process. By introducing a flag data item to mark the updates that are propagated at the current synchronization instant n, we find that the number of the recent updates that should be propagated on the next instant $n+1$ equals to the number of unmarked updates at n, which represent the recent updates that occur after n. Therefore, the value of $U(n+1)$ depends only the value of $U(n)$ and not on any past states.

By using SPN to model the synchronization system (see figure 3 and tables 1-3), the system is composed of eleven places and fourteen transitions.

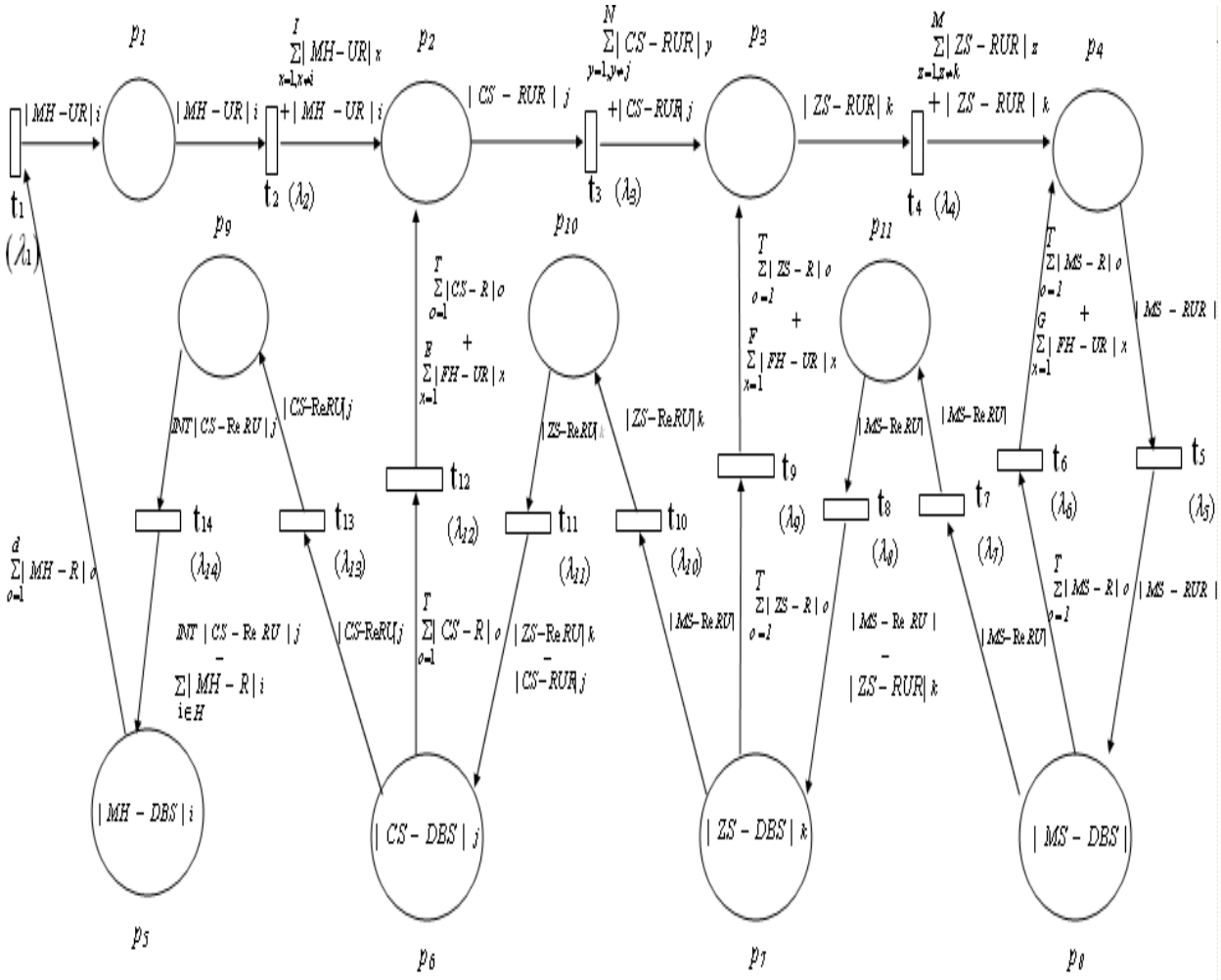

FIGURE 3: The SynchSPN

Place	Description
p_1	IIRA-Synch-State in MH_i for execution in CS_j
p_2	IIRA-Synch-State in CS_j for execution in ZS_k
p_3	IIRA-Synch-State in ZS_k for execution in MS
p_4	IIRA-Synch-State in MS for execution in MS
p_5	DB-Synch-State in MH_i
p_6	DB-Synch-State in CS_j
p_7	DB-Synch-State in ZS_k
p_8	DB-Synch-State in MS
p_9	IIRA-Synch-State in CS_j for execution in MH_i
p_{10}	IIRA-Synch-State in ZS_k for execution in CS_j
p_{11}	IIRA-Synch-State in MS for execution in ZS_k

TABLE 1: Description of Places

Places. They are called synchronization states. The research looks abstractly at the synchronization state as a state/place that contains a set of updates. There are three types of places:

- **IIRA-Synch-State for execution in the higher level:** It contains the set of recent updates that occurred in its host in addition to the set of collected updates from underlying level. For example, the synchronization state p_2 represents the set of all recent updates that issued on cell server *j* in addition to the set of all updates that are transferred from mobile hosts that are synchronized with this server in the last synchronization period. This type includes p_1, p_2, p_3, and p_4. Note that the set of all recent updates that occurred in the servers that exist in the fixed network are assumed that they include also the recent updates that are received from the fixed hosts in the level of those severs.
- **IIRA-Synch-State for execution in the lower level:** It contains the set of all recent resolved updates that are received from the higher level. For example, the synchronization state p_{10} for zone server k represents the set of all recent resolved updates that are received from the master server. This type includes p_9, p_{10}, and p_{11}.
- **DB-Synch-State:** This type stores the set of all tuples that are currently stored in the replicated database. It includes p_5, p_6, p_7, and p_8.

Transition	Description
t_1	Retrieving recent updates from the replicated database in MH_i
t_2	Execution of MHR-IIRA instance on CS_j
t_3	Execution of CSR-IIRA instance on ZS_k
t_4	Execution of ZSR-IIRA instance on MS
t_5	Execution of MSR-IIRA instance on MS
t_6	Retrieving recent updates from the replicated database in MS
t_7	Retrieving recent resolved updates from the replicated database in MS
t_8	Execution of MSR-IIRA instance on ZS_k
t_9	Retrieving recent updates from the replicated database in ZS_k
t_{10}	Retrieving recent resolved updates from the replicated database in ZS_k
t_{11}	Execution of ZSR-IIRA instance on CS_j
t_{12}	Retrieving recent updates from the replicated database in CS_j
t_{13}	Retrieving recent resolved updates from the replicated database in CS_j
t_{14}	Execution of CSR-IIRA instance on MH_i

TABLE 2: Description of Transitions

Transitions. Also, the research looks abstractly at the transition as an event that leads to either retrieving or propagating the set of recent updates. There are three types of transitions: execution of the IIRA instance on the higher level, execution of the IIRA instance on the lower level, and retrieving of recent updates.

Execution of the IIRA instance on the higher level. This type inserts the contents of synchronization state for IIRA for execution in the higher level in the database of the IIRA type that inhabits the higher level. Some conditions must be satisfied in order to fire this type of transitions. These conditions are:

1. Enabling condition. It is composed of two conditions as follows.
    i. There is at least one recent update in the input place of the execution transition.
    ii. The connection with the other host should happen.

The waiting time for the occurrence of the connection is not considered in the period that is required for firing transitions. This is because as previously mentioned, the waiting process for the connection is not considered as a required IIRA's operation for updates propagation. Moreover, the waiting time has a random value for MHs and a deterministic value for the

servers in the fixed network. Therefore, the research interests on the occurrence of the connection as a required condition for firing.

2. Completion of the updates conflicts resolution through ordering process for the collected updates.
3. Migration of the IIRA instance to the other host for execution of its parent synchronization state in that host.
4. Getting the permission for execution in the other host.

Symbol	Meaning
MH, *FH*, *CS*, *ZS, and MS*	Mobile Host, Fixed Host, Cell Server, Zone Server, and Master Server, respectively
T	Total number of database objects
d	Total number of database objects that are replicated in MH_i (d<T)
$\mid X \mid$	The number of updates in the set *X*
MH-R, *CS-R*, *ZS-R*, and *MS-R*	Set of recent updates for object *O* in MH_i, CS_j, ZS_k, and *MS*, respectively
MH-UR, *FH-UR*	Set of recent updates for all replicated objects in MH_i and FH_l, respectively
CS-RUR, *ZS-RUR*, and *MS-RUR*	Set of resolved recent updates at CS_j, ZS_k, and *MS*, respectively
MS-ReRU, *ZS-ReRU*, and *CS-ReRU*	Set of recent resolved updates that are propagated to underlying level from *MS*, ZS_k, and CS_j, respectively
MS-DBS, *ZS-DBS*, *CS-DBS*, and *MH-DBS*	Replicated database state in *MS*, ZS_k, CS_j, and MH_i, respectively
I	Total number of mobile hosts that have synchronized with CS_j before the synchronization of MH_i during the CS_j updates collection period
N	Total number of cell servers that have synchronized with ZS_k before synchronization of CS_j during the ZS_k updates collection period
M	Total number of zone servers that have synchronized with *MS* before synchronization of ZS_k during the *MS* updates collection period
E, F, G	Total number of fixed hosts that have synchronized with CS_j, ZS_k, and *MS*, respectively, during their updates collection period
H	Set of recent updates that are propagated from MH_i to CS_j in the last synchronization period

TABLE 3: Notations and Their Meanings

Each transition from this type can fire in a time instance equals to T^+ that is reached after elapsing of a time period of length T^{up}. The value of T^{up} consists of the time period that is required for ordering the collected updates (Ot^+), the time period that is required for IIRA instance to migrate to the higher level host (Mt^+), and the time period that IIRA instance takes for waiting to get the permission for execution in the higher level host (Wt^+). Since, each one of these values is not deterministic; this means that T^{up} is a random variable. Here, we omit the time that is required for creating the instance, because it is performed locally.

Execution of the IIRA instance on the lower level. This type inserts the contents of synchronization state for IIRA for execution on the lower level in the replicated database of the host that inhabits the lower level. For firing this type, the same conditions that should be satisfied for the first type are applied here, excluding the completion of the updates ordering process.

Each transition from this type can fire in a time instance equals to T^{-} that is reached after elapsing of a time period of length T^{down}. The value of T^{down} consists of the time period that is required for IIRA instance to migrate to the lower level host (Mt^{-}) and the value of the period that IIRA instance takes for waiting to get the permission for execution in the lower level host (Wt^{-}). Also, the time that is required for creating the instance is omitted, since it is performed locally in the same host.

Retrieving recent updates. This type involves either retrieving recent updates for synchronization with the higher level (propagating it to the higher level) or retrieving recent resolved updates for synchronization with the lower level. For firing this type, the enabling condition of the first type should be satisfied.
Each transition from this type can fire in a time instance equals to T^{r} that is reached after elapsing of a time period of length T^{ret}. The value of T^{ret} represents the time period that is required for IIRA to retrieve either recent updates or recent resolved updates from the replica. This value depends on the number of updates that should be retrieved in each synchronization process. Therefore, T^{ret} is a random variable. The periods T^{up}, T^{down}, and T^{ret} represent random variables because their values are obtained depending on non deterministic values.

Note that instead of adding a transition for representing the migration and its associated input place for representing the migrated state, we incorporate them into the execution transition and in the synchronization state, because the migrated state represents the synchronization state itself and the execution of the migration transition encompasses that the synchronization state is already migrated to the other host. Thus, incorporation is performed to prevent the complexity of SynchSPN.

Firing rates. Each transition in SynchSPN is associated with a firing rate (i.e. the parameter λ). This is because the periods T^{up}, T^{down}, and T^{ret} that represent the firing delays after correspondence transitions are enabled are random variables and are assumed exponentially distributed.

The initial marking. In this marking (i.e. m_0), the replicated databases that are stored in the servers in the fixed network (i.e. CS_j, ZS_k, and MS) are assumed identical. Also, the mobile database that is stored in the MH_i is assumed that has received a set of last updates. This received set may represent either all or a subset of the last updates, which occurred or propagated to the fixed network in the period that precedes the time of the last synchronization of MH_i with the fixed network (i.e. during the disconnection time before the time of the last synchronization), which equals to MH_i-$SynchT_{n-1}$ - MH_i-$SynchT_{n-2}$, where MH_i-$SynchT_{n-1}$ is the time of the last synchronization of MH_i with the fixed network and MH_i-$SynchT_{n-2}$ is the time of the synchronization that precedes the last synchronization. Thus, according to the time of the last synchronization (i.e. MH_i-$SynchT_{n-1}$), the mobile database in MH_i is assumed to be in CA state in the marking m_0 if it contains all recent resolved updates. And if for each marking m^{-} reachable from m_0, the mobile database contains a set of recent updates, this means that m^{-} is equivalent to m_0.

4.2 System Behavior

The system behavior (i.e. evolution in time or dynamic changing of markings) is simulated by firing of transitions. The mechanism of firing in SynchSPN is based on the type of transition as follows.

- If t represents the execution of IIRA instance on the higher level, then t will remove the updates that exist in the input place and add them to the previously accumulated recent updates on the synchronization state of the IIRA in the higher level. The accumulated updates represent the updates received from other underlying hosts before the execution of IIRA instance on the higher level in the same time period for updates collection.
- If t represents the execution of IIRA instance on the lower level, then t will remove the recent resolved updates that exist in the input place and add them to the synchronization state of the

replicated database of the host in the lower level and this happens after removing the set of updates that are propagated from the host in the lower level and are included in the set of the recent resolved updates. This is to avoid storing same updates once again.

- If *t* represents the retrieving of recent updates, then *t* will take a snapshot of the recent updates from the input place and add them to the synchronization state for IIRA for execution in either the lower or higher level. This transition does not remove the recent updates from the input source, which represent the synchronization state of the replicated database according to the fact that the retrieving process does not change the state of the database.

Note that when the connection takes place between any two hosts, the firing of the transition that represents the retrieving of recent updates always occurs before the firing of the other two types. This is because updates should be retrieved first before propagation them to the other host.

Based on the initial marking and the firing mechanism, the changing of the markings of SynchSPN is tracked starting from the time of the current synchronization of MHi with the fixed network, which is denoted by MH_i-$SynchT_n$ and ending with the time of the next synchronization, which is denoted by MH_i-$SynchT_{n+1}$. For obtaining a set of last updates, MH_i-$SynchT_{n+1}$ should occur in this tracking after firing of all transitions. The tracking of marking changing is also depends on the fact that the MH_i will obtain the last updates that are performed on both fixed and mobile networks only after propagating these updates from their sources to the higher levels , where these updates are resolved. Therefore, bottom-up propagation is considered first then the top-down propagation. Thus, the firing sequence of the transitions is divided into two sequences as shown in table 4.

Propagation type	Firing sequence
Bottom-Up	$t_1 \rightarrow t_2 \rightarrow t_{12} \rightarrow t_3 \rightarrow t_9 \rightarrow t_4 \rightarrow t_6 \rightarrow t_5$
Top-Down	$t_7 \rightarrow t_8 \rightarrow t_{10} \rightarrow t_{11} \rightarrow t_{13} \rightarrow t_{14}$

TABLE 4: The Firing Sequence of the Transitions

According to the specified firing sequence, the set of all reachable markings from m_0 are shown in table 5. This set represents the evolution of the system in the period MH_i-$SynchT_{n+1}$ - MH_i-$SynchT_n$. The marking that reachable from firing of t_{14} represents the marking in which the replicated database in MH_i should obtain a set of recent updates that occurred or propagated to the fixed network during that period. This means that this marking is equivalent to m_0.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}
m_0	0	0	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|$	0	0	0
m^-_0	$\|MH\text{-}UR\|_i$	0	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|$	0	0	0
m_1	0	$T_{MH\text{-}CS}$	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|$	0	0	0
m^-_1	0	$\|CS\text{-}RUR\|_j$	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|$	0	0	0
m_2	0	0	$T_{CS\text{-}ZS}$	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|$	0	0	0
m^-_2	0	0	$\|ZS\text{-}RUR\|_k$	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|$	0	0	0
m_3	0	0	0	$T_{ZS\text{-}MS}$	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|$	0	0	0
m^-_3	0	0	0	$\|MS\text{-}RUR\|$	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|$	0	0	0
m_4	0	0	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|+$ $\|MS\text{-}RUR\|$	0	0	0
m^-_4	0	0	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k$	$\|MS\text{-}DBS\|+$ $\|MS\text{-}RUR\|$	0	0	$\|MS\text{-}ReRU\|$
m_5	0	0	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k +$ $T_{MS\text{-}ZS}$	$\|MS\text{-}DBS\|+$ $\|MS\text{-}RUR\|$	0	0	0
m^-_5	0	0	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j$	$\|ZS\text{-}DBS\|_k +$ $T_{MS\text{-}ZS}$	$\|MS\text{-}DBS\|+$ $\|MS\text{-}RUR\|$	0	$\|ZS\text{-}ReRU\|_k$	0
m_6	0	0	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j +$ $T_{ZS\text{-}CS}$	$\|ZS\text{-}DBS\|_k +$ $T_{MS\text{-}ZS}$	$\|MS\text{-}DBS\|+$ $\|MS\text{-}RUR\|$	0	0	0
m^-_6	0	0	0	0	$\|MH\text{-}DBS\|_i$	$\|CS\text{-}DBS\|_j +$ $T_{ZS\text{-}CS}$	$\|ZS\text{-}DBS\|_k +$ $T_{MS\text{-}ZS}$	$\|MS\text{-}DBS\|+$ $\|MS\text{-}RUR\|$	$\|CS\text{-}ReRU\|_j$	0	0
m_7	0	0	0	0	$\|MH\text{-}DBS\|_i +$ $T_{CS\text{-}MH}$	$\|CS\text{-}DBS\|_j +$ $T_{ZS\text{-}CS}$	$\|ZS\text{-}DBS\|_k +$ $T_{MS\text{-}ZS}$	$\|MS\text{-}DBS\|+$ $\|MS\text{-}RUR\|$	0	0	0

TABLE 5: The Marking Table

In the marking table, the marking m_7 is equivalent to m_0 because the former represents the state in which the mobile database in MHi receives a set of recent updates that occurred or propagated to the fixed network during the period: $MH\text{-}Synch T_{n+1}$ - $MH\text{-}Synch T_n$

The marking table includes the following Equations:

$$T_{MH\text{-}CS} = \sum_{x=I, x\neq i}^{I} |MH - UR|_x + |MH\text{-}UR|_i \quad (1)$$

Where $T_{MH\text{-}CS}$ is the total number of updates that will be propagated to *CSj* during its updates collection period form *I* mobile hosts.

$$|CS\text{-}RUR|_j = T_{MH\text{-}CS} + |CS\text{-}UR|_j \quad (2)$$

This equation represents the total number of resolved updates that will be propagated from CSj to ZSk

$$T_{CS\text{-}ZS} = \sum_{y=I, y\neq j}^{N} |CS - RUR|_y + |CS\text{-}RUR|_j \quad (3)$$

Where $T_{CS\text{-}ZS}$ is the total number of updates, which will be propagated to *ZSk* during its updates collection period form *N* cell servers.

$$|ZS\text{-}RUR|_k = T_{CS\text{-}ZS} + |ZS\text{-}UR|_k \quad (4)$$

This equation represents the total number of resolved updates that will be propagated from ZSk to MS.

$$T_{ZS\text{-}MS} = \sum_{z=1, z\neq j}^{M} |ZS - RUR|_z + |ZS\text{-}RUR|_k \quad (5)$$

Where $T_{ZS\text{-}MS}$ is the total number of updates that will be propagated to *MS* during its updates collection period form M zone servers.

$$|MS\text{-}RUR| = T_{ZS\text{-}MS} + |MS\text{-}UR| \quad (6)$$

This equation represents the total number of resolved updates that will be stored in the database that exists in MS.

$$T_{MS\text{-}ZS} = |MS\text{-}ReRU| - |ZS\text{-}RUR|_k \quad (7)$$

Where $T_{MS\text{-}ZS}$ is the total number of resolved updates that will be propagated to ZS_k database from *MS* excluding updates that previously propagated from ZS_k

$$T_{ZS\text{-}CS} = |ZS\text{-}ReRU|_k - |CS\text{-}RUR|_j \quad (8)$$

Where $T_{ZS\text{-}CS}$ is the total number of resolved updates that will be propagated to CSj database from ZSk excluding updates that previously propagated from CSj.

$$T_{CS\text{-}MH} = |CS\text{-}ReRU|_k - |MH\text{-}UR|_j \quad (9)$$

Where $T_{CS\text{-}MH}$ is the total number of resolved updates that will be propagated to MHi database from CSj excluding updates that previously propagated from MHi.

Reachability graph. This graph is described in figure 4. The markings m_i^- , where i=0,1,…,6 that reachable from firing of retrieve transitions are not included in the reachability graph because this type of transitions affects only the local synchronization state for IIRA database. However, these transitions are included, since their firing precedes the firing of the execution transitions that leads to the markings m_j, where j=1, 2,…, 7.

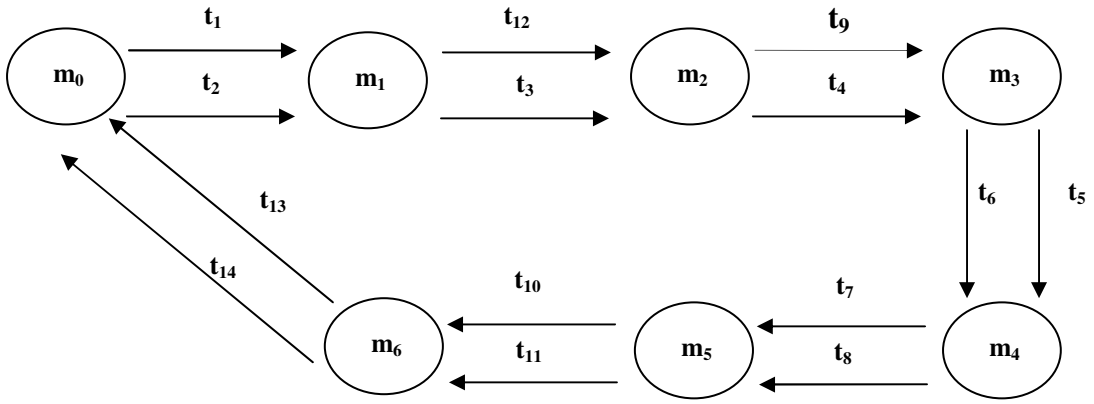

FIGURE 4: Reachability Graph

Equivalent Markov chain. In the derived CTMS (see figure 5), the firing rates of the retrieve transitions are not considered because as mentioned previously, the retrieve operation is performed locally from the replicated database on a given host.

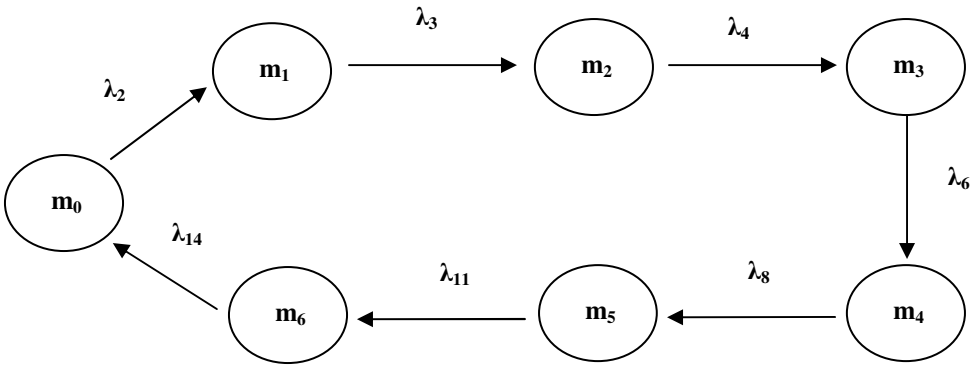

FIGURE 5: Derived Markov Chain

Analysis of the Markov chain. The steady-state probabilities, denoted by $\prod = (\pi_0, \pi_1, \pi_2, \ldots, \pi_6)$ are obtained by solving the following equations:

$$\prod A = 0 \qquad (10)$$

$$\sum_{i=0}^{6} \pi_i = 1 \qquad (11)$$

Where A is the transition rate matrix. π_i is the steady-state probability of marking that is equivalent to m_i.

The obtained matrix for the derived Markov chain is shown in figure 6.

	m_0	m_1	m_2	m_3	m_4	m_5	m_6
m_0	$-\lambda_2$	λ_2	0	0	0	0	0
m_1	0	$-\lambda_3$	λ_3	0	0	0	0
m_2	0	0	$-\lambda_4$	λ_4	0	0	0
m_3	0	0	0	$-\lambda_6$	λ_6	0	0
m_4	0	0	0	0	$-\lambda_8$	λ_8	0
m_5	0	0	0	0	0	$-\lambda_{11}$	λ_{11}
m_6	λ_{14}	0	0	0	0	0	$-\lambda_{14}$

FIGURE 6: Transition rate matrix

By solving Eq. 1 and Eq. 2, the obtained steady-state probabilities as follows:

$$\Pi = \begin{pmatrix} \pi_0 \\ \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \end{pmatrix} = \begin{pmatrix} 1/(1+\lambda_2\omega_0) \\ 1/(1+\lambda_3\omega_1) \\ 1/(1+\lambda_4\omega_2) \\ 1/(1+\lambda_6\omega_3) \\ 1/(1+\lambda_8\omega_4) \\ 1/(1+\lambda_{11}\omega_5) \\ 1/(1+\lambda_{14}\omega_6) \end{pmatrix}$$

Where $\omega_0 = 1/\lambda_3+1/\lambda_4+1/\lambda_6+1/\lambda_8+1/\lambda_{11}+1/\lambda_{14}$, $\omega_1 = \omega_0 -(1/\lambda_3+1/\lambda_2)$, $\omega_2 = \omega_0 -(1/\lambda_4+1/\lambda_2)$, $\omega_3 = \omega_0 -(1/\lambda_6+1/\lambda_2)$, $\omega_4 = \omega_0 -(1/\lambda_8+1/\lambda_2)$, $\omega_5 = \omega_0 -(1/\lambda_{11}+1/\lambda_2)$, $\omega_6 = \omega_0 -(1/\lambda_{14}+1/\lambda_2)$
As previously mentioned, the research interests in the state in which the mobile database in the mobile host contains a set of recent updates. Therefore, the value of π_0 represents the probability of the marking that is equivalent to m_0.

Assertion 4.2.1. There is only a subset $C \subseteq R(m_0)$, such that the mobile database in specific mobile host in CA State for each $m \in C$.
Proof. Let t_j ($j=1,\ldots,m$) denotes the transition that represents the execution state of the CSR-IIRA in MH_i, and t_i ($i=1,\ldots,n$) denotes the transition that represents the execution state of the MHR-IIRA in the fixed network. We show that firing of t_j will result in CA State for R that is hosted in $MHi \Leftrightarrow$ each t_i is fired during the time period that precedes the current synchronization time of MH_i. Since the latter condition is not realized at all synchronization times for MH_i due to existing of many MHs are not connected before the synchronization of MHi with the cell server. Therefore, the firing of t_j leads to CA State $\Leftrightarrow$ all updates that are performed in the mobile network are propagated to the fixed network before the synchronization of MH_i. This means that if the latter condition is realized, the firing of t_j will results in a marking that represents an element of C.

Assertion 4.2.2. The probability that the mobile database in CA state is:

$$P(\text{CA}) = n\pi_0 \qquad (12)$$

Where n is the number of synchronization times for MH_i with the fixed network that led to the CA state.
Proof. Let C be the subset of $R(m_0)$ satisfying the condition that the place p_5 has received all recent updates that occurred and resolved before the synchronization time of MH_i with the fixed network, which led to each marking in C. Then the probability of this condition is: $P(C) = \sum_{i \in C} \pi_i$,
Since the probability that MH_i receives a set of recent updates is π_0. Then $\pi_i = \pi_0$ for each marking $m_i \in$ C. This means that $P(C) = \sum_{i=1}^{n} \pi_i = n\pi_0$, where n is the number of markings in C. This number is equivalent to the number of the synchronization times with the fixed network that led to storing all recent updates in p_5.
Assertion 4.2.3. The marking m_0, where p_5 receives a set of recent resolved updates is recurrent after a time period equals to: $MH_i\text{-}SynchT_{n+1} - MHi\text{-}SynchT_n$
, where $MHi\text{-}SynchT_{n+1} - MHi\text{-}SynchT_n > \lambda_5 + \lambda_8 + \lambda_{11}$, $MHi\text{-}SynchT_n$ is the time instant of the current synchronization with the fixed network, $MH_i\text{-}SynchT_{n+1}$ is the time instant of the next synchronization with the fixed network.
Proof. m_0 is recurrent if the following equation is satisfied.

$$\sum_{i=MHi\text{-}SynchTn}^{MHi\text{-}SynchTn+1} P^{i}_{m_0 m_0} = 1 \qquad (13)$$

Where P^{i}_{m0m0} is the probability that the system returns to state m_0 starting from m_0. Suppose that the markings that are reachable from m_0 occur at the following time instants:
$MHi\text{-}SynchT_n, T_1, T_2, \ldots, T_n, MHi\text{-}SynchT_{n+1}$, where $MHi\text{-}SynchT_n < T_1 < T_2 < \ldots < T_n < MHi\text{-}SynchT_{n+1}$. Obviously, the MH_i can obtain the last updates that occurred or are propagated to the fixed network during the period: $MHi\text{-}SynchT_{n+1} - MHi\text{-}SynchT_n$ after a time instant > = $MHi\text{-}SynchT_{n+1}$, which means that:
$P^{MHi\text{-}SynchTn}{}_{m0m0} = 0$, $P^{MHi\text{-}SynchTn+1}{}_{m0m0} = 1$, $P^{T1}{}_{m0m0} = 0$, $P^{T2}{}_{m0m0} = 0$, …, $P^{T0}{}_{m0m0} = 0$
To obtain the latest updates, the following condition should hold:

$$MH_i\text{-}SynchT_{n+1} - MH_i\text{-}SynchT_n > \lambda_3 + (\lambda_4 - \lambda_3) + (\lambda_5 - \lambda_4) + \lambda_8 + \lambda_{11} = \lambda_5 + \lambda_8 + \lambda_{11}$$

Where $\lambda_3 < \lambda_4 < \lambda_5$. This is because the server in the master level receives updates from all underlying levels, while the servers in the zone and cell levels receive updates from the hosts that are located in their areas.

Assertion 4.2.4. The SynchSPN is deadlock free.
Proof. We prove that $\forall\ m \in R(m_0)$, $\exists\ m^{-} \in R(m)$ such that $\exists\ t$ is enabled for m^{-}, where $R(m_0)$ is the set of markings reachable from m_0 by firing a sequence of transitions. Recall that our assumptions regarding that tens of updates per each data item are expected at any period of time and the connection is reliable and fixed between the servers of the fixed network, this means that the following conditions are true:

1. At any period of time, $\exists\ DB \in RDB$ such that *DB* is updated recently (has UR) where *RDB* is the set of the replicated databases in either fixed or mobile hosts. This condition ensures enabling of selection transactions.
2. There is at least one host (either fixed or mobile) is synchronized with other host (e.g. MH with a cell server or other MH, FH with a server, CS with ZS…etc). This condition ensures that there is at least one of the execution transitions is enabled after satisfying condition one.

5. CONTRIBUTION AND DISCUSSIONS

The contributions of this paper can be summarized as follows: firstly, a new replication strategy is presented for replicating data by considering a logical three levels architecture and a multi-agent based replication method. 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. Second, the proposed architecture supports scalability by allowing large numbers of updateable replicas in the cell level and higher levels, since the large scale distributed database systems require such a feature. Third, the paper combines both optimistic and pessimistic replication approaches, in a hybrid manner that exploits the pertinent features of each in mobile environments.

The IIRA-dependant multi-agent system achieves load balance in both propagation and ordering processes. This is because these processes are shared by multiple hosts, where each host propagates a set of the recent updates to another in either lower or higher level and each host participates in ordering the updates that are issued in its replicated database or collected from underlying levels.

To decrease the communication cost and provide a better utilization of the connection time in environments prone to more frequent disconnections and failures, the proposed strategy relies on instance immigration instead of the migration of the agent itself as in mobile agents' communities. When the connection takes place, the instance holding the updates-result will migrate to the other host, and performs its task. Then it removes itself without needing to return back as in mobile agents' communities. This minimizes the connection cost to only cost that is needed to transfer one instance per connection time.

The availability of all recent updates that occurred in both fixed and mobile networks to the mobile hosts depends directly on the propagation of these updates from their sources to the fixed network. This propagation should takes place before the synchronization of the mobile host with the fixed network.

The replication method ensures achieving the consistency of data in the mobile network according to the time of last connection happened. Therefore, data inconsistency here will depend on the difference between the connection times.

To minimize the rate of update conflicts, and taking a step on ensuring eventual consistency, data are divided into two types: infrequently changed data that are updated only in the master level and frequently changed data that are updated on the underlying levels.

6. CONCLUSTIONS

In this paper, our research has focused on proposing a new replication strategy to maintain consistency and improve availability of data in large scale mobile environments. The replication strategy encompassed replication architecture and replication method as a binary combination that is needed to achieve such a goal. To exploit the features of both optimistic and pessimistic replication, the new strategy is based on a hybrid approach that divides data into frequently changed data and infrequently changed data, and then updates are restricted or allowed according to these types. Stochastic Petri Net is developed to model the dynamic behavior of the replicated system in regard to reaching the Consistent-Available state for the replicated database at the mobile host.

As a part of our future research, a plan will be provided to develop the required tools and interfaces to implement the proposed strategy in mobile healthcare environments to provide healthcare practitioners with an efficient access to healthcare data.

7. REFERENCES

1. T. Connolly and C. E. Begg. "*Database Systems: A Practical Approach to Design, Implementation and Management*". 4th edition, Addison-Wesley, (2004)

2. S. Madria and S. Bhowdrick. "*Mobile data management*". Potentials, IEEE, 20(4):11 – 15, 2001

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

4. Y. Saito and M. Shapiro. "*Optimistic replication*". ACM Computing Surveys (CSUR), 37(1):42–81, 2005

5. J. Gray, P. Helland , and P. O'Neil. "*The dangers of replication and a solution*". In Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 173–182, 1996

6. M. Wiesmann, A. Pedone, B. Schiper, G. Kemme and Alonso. "*Understanding replication in databases and distributed systems*". In Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS 2000), pp. 464, 2000

7. P. Bernstein. "*Principles of Transaction Processing*". Morgan Kaufmann Publishers Inc, (1997)

8. A. Helal, A. Heddaya and B. Bhargava. "*Replication Techniques in Distributed Systems*". Kluwer Academic Publishers, 1996

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

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

11. J. Monteiro, A. Brayner and S. Lifschitz. "*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

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

13. N. Tolia, M. Satyanarayanan and A. Wolbach. "*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, 2007

14. G. Balbo. "*Introduction to generalized stochastic Petri Nets*". Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 4486/2007: 83-131, 2007

15. G. Balbo., "*Introduction to stochastic Petri Nets*" Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 2090/2001, pp. 84-155, 2001

16. M. Ajmone, G. Balbo, G. Conte, S. Donatelli and G. Franceschinis. "*Modeling with Generalized Stochastic Petri Nets*". J. Wiley, Chichester. (1995)

17. M. Ajmone, G. Balbo and G. Conte. "*Performance models of multiprocessor systems*". MIT Press, Cambridge, 1986

18. F. Bause and P. Kritzinger. "*Stochastic Petri Nets -- An Introduction to the Theory*," 2nd edition, Springer Verlag, Germany, 2002

19. T. Murata. "*Petri Nets: properties, analysis and applications*". In Proceedings of the IEEE, pp.541–580, 1989