CHAPTER 4
MULTI AGENT BASED REPLICATION FRAMEWORK FOR THE MOBILE GRID ENVIRONMENT

4.1 INTRODUCTION

Mobile devices have been recently researched on the distributed and grid systems for collaboration and task execution (Grabowski et al 2006). Due to the transient nature of wireless devices, its integration into grid environment is a challenging task. The successful adoption of replication in the mobile grid computing relies on middleware services which handles infrastructure effectively. Such realization is primarily affected by the following key issues,

- Device capabilities

  - With the continuous progress in the development of mobile devices, a wide variety of devices exist with improving capabilities. However, compared to the desktops, mobile devices are resource-limited. Middleware should have mechanisms for resource and energy optimization procedures.

- Network Connections

  - In the mobile environment, network connections usually have limited bandwidth and frequent disconnections.
Middleware should find reliable site by considering the network characteristics such as link quality (Signal-to-Noise ratio) for task execution and to hold copy of the data in order to improve the performance of the environment.

- **Dynamic Characteristics**
  
  - In the dynamically changing mobile grid, the environment should be configured automatically to satisfy the requirement of multiple users. Autonomic behaviors such as self management, self optimization and self configuration are desired properties to support the dynamic environment. These kinds of services are to be reconfigured at run time, say, to adapt the amount of memory available for their use, number of users accessing the same data file and relocation of the data file.

  The existing grid infrastructure and architecture do not consider the design and implementation of the integrating mobile systems. However, integrating mobile devices into the grid environment brings considerable attention among the recent researchers.

  To address these issues, a multi agent based framework is proposed to implement intelligent interaction mechanism between the grid middleware and mobile devices. The following section briefs about the multi agent systems and its advantages.

**4.2 AGENT BASED SYSTEM**

Software Agent is a piece of code that functions continuously and autonomously in a particular environment, which may contain other agents
and processes. The idea is that agents are not strictly invoked for a task, but activate themselves. The different types of agents are as follows,

- **Intelligent agents**
  - Exhibiting some aspect of Artificial Intelligence, such as learning and reasoning

- **Autonomous agents**
  - Capable of modifying the way in which they achieve their objectives

- **Distributed agents**
  - Being executed on physically distinct computers

- **Multi-agent systems**
  - Distributed agents that do not have the capabilities to achieve an objective alone and thus must communicate

- **Mobile agents**
  - Agents that can relocate their execution onto different processors

### 4.2.1 Advantages of Agents

In the dynamically changing mobile grid environment, autonomic behaviors such as self management, self optimization and self configuration are the desired properties to satisfy the requirements of the users. Such behaviors should adapt to the environmental properties, in order to meet the
system design objectives. The following advantages motivate to adopt multi-agent based computing mechanism to handle the issue of autonomic behaviors in the mobile grid environment.

- Reducing network load
  - Agents on the mobile node and base station reduce the number of message exchanges between the mobile node and the base station, thus helping to reduce the communication cost of data access.

- Executing asynchronously and autonomously
  - The agent can operate autonomously even if the node from where it was launched is no longer available.

- Overcome network latency
  - Mobile agents dispatched from the base station act locally and directly execute the instructions thus helping in overcoming the network latency in achieving replica consistency across the sites.

- Adapting dynamically
  - The agents can be retracted, dispatched, cloned, or put to sleep as the network, and host conditions change. For example, as better agents are developed, they can be sent out on the network to replace the older version.
• Operating in heterogeneous environments
  
  o The agents are dependent only on their execution environment, they facilitate heterogeneous system integration. This advantage is vital in the mobile environment due to device heterogeneity (i.e., smart phones, PDA’s and laptops are involved in mobile grid formation).

• Robust and fault tolerant
  
  o Mobile agents can react dynamically to unfavorable changes in the environment which helps to create a robust and fault-tolerant grid system.

4.3 ARCHITECTURE DESIGN OVERVIEW

One of the main design principles of the distributed system design is transparency. The definition of transparency implies openness, communication and accountability. In a distributed architecture such as grid and mobile grid systems, transparency refers to hiding unimportant implementation details from the system users. The middleware in the system architecture is required to provide a transparent access mechanism enabling mobile devices to take part as grid resources while shielding them from the complex system implementation details.

The proposed framework provides a comprehensive infrastructure for the improving the data availability and supporting a small number of replicas in the mobile grid environment by determining the required components which are involved in the replication process. Also, it couples job scheduling with the knowledge of data replication. The layered description of the proposed system is depicted in the Figure 4.1.
At the infrastructure level, the grid environment shares resources between the distributed users and sites and it can be fixed node clusters, workstations and mobile nodes (laptop).

Job scheduling and data replication are two effective techniques to enhance the performance of the grid environment. Job scheduling is to map and dispatch a set of jobs on to sites for execution and to minimize the job execution time. Each job usually requires multiple input files for its execution. Therefore, when scheduling a job at a site, the data transfer time of its input file has to be taken into consideration. On the other hand, data replication creates multiple copies of data in the environment to reduce the data transfer time and bandwidth consumption, and to minimize the total communication cost.
These two techniques are complementary with each other. Scheduling without considering replication places an overhead of data transfer time as the jobs input files have to be fetched remotely. However in a resource restricted mobile grid environment, integrating job scheduling and data replication into one framework can minimize the communication overhead.

In order to realize this, the proposed design includes data management functionalities such as selecting reliable replication, replica creation and relocation.

Moreover, to reduce the complexity of the resource sharing in these highly dynamic and heterogeneous scenarios, new components and software layers are required. The software component responsible for the resource management and scheduling is usually called the grid resource broker or meta-scheduler. The meta-scheduler plans for assigning the job to the particular sites among the available resources. In such a situation, dynamic characteristics of the mobile grid system such as battery lifetime, load on the nodes and connectivity are to be considered. In order to support the meta-scheduler in this context an intelligent agent based framework is proposed.

With respect to the system design and implementation of the system, transparency is achieved in two specific aspects,

- Agent adaptation
  - Communication overhead in wireless network should be managed effectively. The proposed architecture uses multiagent based approach to intelligently manage the resources, replica and consistency in the resource constrained environment.
• Reliable task achievement

  o Mobile devices are characterized by their limited resources (battery) and intermittent connectivity (Signal quality due to mobility), which leads to an unreliable availability in the grid environment. This makes it very difficult to manage task execution on mobile devices. The middleware has to provide a reliable management mechanism to execute the tasks submitted to the mobile grid environment.

4.4 SYSTEM ARCHITECTURE IMPLEMENTATION

Building a transparent intelligent interaction mechanism requires middleware systems to have the ability of understanding the context information of the computing environment. In such situations, grid resources need to share their knowledge with the scheduler, acquiring various context information of the system and support autonomic behaviors. In order to achieve this, the proposed framework considers the infrastructure as depicted in the Figure 4.2.

In the Figure 4.2, the mobile grid system is divided into three parts, namely Static grid sites, group of mobile devices and a gateway interconnecting static and mobile resources. The resource associated with each mobile node is having enough processing power. Mobile devices are wirelessly connected to the internet through a wireless LAN or a long range cellular network such as Wi-Max and CDMA. Here, both the fixed and mobile nodes are used for task execution and data management. The middleware Globus Tool Kit version 4 is installed on the gateway to coordinate the scheduling among the mobile sites and fixed grid sites.
The resource management such as task execution and data replication at the middleware level is taken care by the meta-scheduler. Meta-scheduler gains knowledge about the availability and status of the resources using information services. Based on this, the task will be scheduled to the available sites. To support the meta-scheduler to handle dynamic characteristics of mobile devices, different types of agents adapted in the framework at different levels and the different functional components of agents are discussed in next section.
4.4.1 Types of Agents

Agents used in the framework are classified into three types according to the level in which the agents carry out their activities and functionality that are given below,

- BS Agent (Base Station Agent)
  
  BS-Agent is responsible for analyzing the context of the environment. It analyzes the system requirement and detects any changes in the grid environment and reports to scheduler.

- N Agent (Node Agent)
  
  N-Agent is responsible for monitoring the mobile nodes and collecting mobility, battery, load, SNR and SNRc information and maintains the read / update counter for replica.

- U Agent (Update Agent)
  
  U-Agent is responsible for updating the mobile node replica and maintains consistency across the mobile grid.

4.4.2 Functional Components

To facilitate the meta-scheduler for handling the mobile devices in the grid environment, some of the functional components have been identified and the interactions between these components are depicted in the Figure 4.3. The main modules of the proposed framework are listed,
- Context analyzer
- System state monitor
- Strategy manager
- Localization manager
- Consistency manager
- Replica manager

Figure 4.3 Functional Components
- **Replica manager:**
  - The replica manager selects the optimum number of locations to place the replica such that the maintenance overhead includes the update propagation, access cost and storage cost, that can be minimized with minimum number of replications.

- **Localization manager:**
  - The localization manager keeps track of the nodes containing the replica and is responsible for update propagation.

- **Consistency manager:**
  - The consistency manager ensures replica consistency after each write operation and resolving update conflicts.

- **Context analyzer:**
  - The context analyzer detects the pertinent change in the required context information and provides the context information.

- **Strategy manager:**
  - The strategy manager adapts the replication strategy to the context or system state variations.

- **System state monitor:**
  - Replication system monitors its own state. This state is represented by the system performance (load) parameters, data availability parameters, and data consistency parameters.
4.4.2 Agent States

The possible states of the proposed agents are,

- Monitoring
  - Monitor the connection with the other mobile devices through interaction with environment via message passing.

- Tokenizer
  - The environment change and data update list is passed to the base station via message passing.

- Creating instance
  - The U-Agent creates an instance of it and stores the recent updates on this instance.

- Migration
  - The U-Agent instance migrates from the base station to the other mobile devices which contain replica that is inconsistent.

- Insertion
  - In this state, the U agent instance inserts its stored recent updates in the data set of the mobile device.

- Removing
  - The migrated instance of U-Agent removes itself after completion of the insertion process.
4.5 IMPLEMENTATION

The maintenance of the replication using the proposed framework on a mobile grid environment is implemented at two levels namely,

- Base Station Level and
- Mobile Host Level.

In each level the different functional components defined above are integrated with the agents and the algorithms as explained below.

4.5.1 Base Station Level

This level contains the master replica, which must be synchronized with the replicas from the nodes at the mobile host level. At this level, two types of proposed agents are associated and the replica manager module is integrated.

Replication manager

The replica manager selects the optimum number of locations to place the replica such that the maintenance overhead such as update propagation, access cost and storage cost can be minimized with the minimum number of replications. It considers the following information to place and relocate replica dynamically.

- Signal – to – Noise Ratio
- Battery information
- Mobility information
- Storage information
- Load information
**Signal – to – Noise Ratio**

The Signal-to-Noise ratio referred as SNR is broadly defined as the ratio between the maximum signal strength that a wireless connection can achieve and the noise present in the connection. SNR of a network needs to be as high as possible. SNR will be measured in units of decibels (dB).

\[
SNR = \frac{V_s}{V_n} \tag{5.1}
\]

Where \( V_s \) is the incoming signal strength in micro volts, \( V_n \) is the noise level, also in micro volts.

**Battery Information**

Mobile host running on limited energy sources (battery) communicate with each other using wireless links.

\[
B_i = B_{ic}-B_{ip} \tag{5.2}
\]

where, \( B_{ic} \) is the current amount of battery before a replica is assigned, \( B_{ip} \) is the amount of battery to transfer data the object from the base station based on the available bandwidth and data object size. If \( B_i \) is less than 20 percent of the total amount of battery, replica is not assigned.

**Mobility Information**

Replica should not be assigned to a node if it has a frequent movement. Replica decision will be made using the preprocessed (history of past locations) periodical mobility patterns and the node’s current mobility information.
Storage Information

Before placing a replica, the available storage $S_i$ of mobile node has to be identified. If the available storage is less than the required storage, the replica will not be placed.

$$S_i = S_{avail} - \text{file size} \quad (5.3)$$

Load Information

In order to reduce the workload of a mobile node, load information on the mobile node is considered before placing the replica. If the CPU of a node is already engaged with heavy task, placing a replica degrades the performance of the system.

With this information, the replication manager makes use of self stabilizing distributed dynamic replication algorithms for data object placement in the mobile grid. The distributed dynamic replication algorithm works by three tests, namely, expansion test, contraction test, and switch test.

The expansion test (Algorithm 1) is executed at each neighbor of i, if the changes like one or more mobile node getting added to the wireless grid, detection of drastic changes in read and update requests of the mobile nodes in a particular instance triggers the execution of expansion test by the BS-Agent and is responsible for replica schema expansion, aging factor can be used to modify the third step to improve the efficiency of the expansion test. It is easier to deal with the prevention of node failures due to communication network than to solve the node failure. The proposed method prevents data replication at nodes which has the tendency towards failure.

The contraction test (Algorithm 2) is done only at the nodes which have the replica. The contraction test is triggered by the detection mobile
node failure by the BS-Agent and is responsible for replica schema contraction. If any such node failure is noticed, the contraction test then allows releasing the site from the replication schema.

The switch test (Algorithm 3) is executed at the nodes which has replica and is responsible for moving the replica to the neighboring node by discarding its own copy. If the candidate node that is holding the replica currently is neither the best site to fetch nor can handle the access request due to the change in user request and network latency, the reallocation of the replica is then done by the switch test execution. The algorithm will converge to the optimal placement scheme if the user access pattern remains stable and all the three algorithms are given below,
Algorithm 1:

1: Begin
2: for Each neighbor $j \in M$ do
3: Examine $R_i$ and $W_i$. $R_i$ is the read request from $j$ to $i$ during last time period, $W_i$ is the total number of update request from $i$ and neighbor $6 = j$ to $i$.
4: if $R_i > W_i$ then
5: Mark those nodes as candidate nodes.
6: end if
7: end for
8: for Each candidate do
9: if Candidate is the site on which replica is currently located then
10: Goto step 21
11: else
12: For the candidate sites get SNR and SNRc, battery, mobility, load values.
13: if SNR > SNRc, battery > 10 %, mobility < 40 % and load < 70 % then
14: Replicate the data on the candidate node (thus, $j$ joins $M$).
15: else
16: Choose the site with the read requests > update requests, from the non candidate set and mark it as candidate for data replication.
17: Go to step 8
18: end if
19: end if
20: end for
21: End
Algorithm 2:

1: Begin
2: for Each node \( i \in M \) do
3: Examine \( W_j \) and \( R_i \). \( W_j \) is the number of update requests that \( i \) received from \( j \) during the last time period, \( R_i \) is the number of read requests that \( i \) received in the last time period.
4: Examine SNR, SNRc, battery, mobility, load values.
5: if \( W_j > R_i \) or \( \text{SNR} \leq \text{SNRc} \) or battery \( < 10 \% \) or mobility \( > 40 \% \) or load \( > 70 \% \) then
6: Release site from replication schema \( M \) (thus, \( i \) exits \( M \))
7: else if The replica node \( i \) doesn’t respond then
8: Release site from replication schema \( M \) (thus, \( i \) exits \( M \))
9: end if
10: end for
11: End
Algorithm 3:

1: Begin

2: if If there is only i in M then

3: for Each neighbor k do

4: Examine two integers denoted by R_k and R_k*. Integer R_k is the number of read requests received by i from k during the last time period, and R_k* is the number of all other read requests received by i during the last time period.

5: Examine SNR, SNR_c, battery, mobility, load values.

6: if R_k > R_k* and if SNR > SNR_c, battery >10%, mobility <40 %, load <70 % then

7: i sends the replica to k with an indication that k becomes the new singleton processor in the replication schema (thus, i exits M & k joins M)

8: end if

9: end for
Localization Manager:

The localization manager keeps track of the nodes containing the replica and is responsible for the allocation of U-Agent upon the update request. The following pseudo code demonstrates the function of the localization manager.

Pseudo Code

```c
struct LocationInfo
{
    AP_Locateinfo; /*Access Point location information*/
    Node_Repinfo; /*Replica node information*/
    Node_Datainfo; /*Replica data information*/
}
do {
    for each (data object) {
        get replica_info from Replication manager;
        allocate location information and store it in history base;
        send LocationInfo to Middleware Gateway;
    }
} while(true);
```
Context Analyzer:

The context analyzer detects the pertinent change in context information such as network status information (SNR), remaining battery power, available storage, location and mobility direction information is transmitted to middleware gateway and analyzed. The gateway then informed the replica manager through the strategy manager to decide about the creation of a replica or relocation necessary. Also, it has to maintain the system stable in case of failures.

Pseudo Code

```
struct Contextinfo
{
    signalInfo; /*SNR Information*/
    bandwidthInfo; /*Bandwidth Information*/
    batteryInfo; /*Available battery Information*/
    loadInfo; /*CPU load Information*/
    ReadCount/*number of read request*/
    WriteCount/*number of write request*/
}
do
    { 
        (for each mobile device) 
        { 
            calculate SNR, required battery and load; 
            calculate available bandwidth; 
            send Contextinfo to middleware gateway; 
        } } while(true);
```
Consistency Manager:

The consistency manager ensures replica consistency after each write operation and resolving update conflicts and creating the instance of U-Agent which is responsible for inserting update to each replica site and helps maintaining consistency.

Pseudo Code

Struct ConsistInfo

{  
    updateList; /*list of updates*/  
    dataObjname; /*data file required to be updated*/  
}

do

{  
    for each update request
    {
        call LocationInfo();
        create update instance on update requesting node;
        remove the update instance;
    }
}

while (true);
**Strategy Manager:**

In order to adopt different replica placement strategies on a mobile grid, framework has strategy manager which can adapt any one replication strategy based on context or system state variations. For example, the system changes its strategy from pessimistic to optimistic in order to improve the data availability. The pessimistic approach is used for restricting the updates of the infrequently changed data to a single replica. 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 once they have occurred (Ashraf A Fadelelmoula et al 2009).

**System State Monitor:**

Replication system monitors its own state. This state is represented by the system performance (load) parameters, data availability parameters, and data consistency parameters. Replication scheme that is most adapted to the change in the context by modifying the plans are stored in history. For example, if the response time increases, the replica planner modifies some replica locations in order to reduce this parameter based on history.

**4.5.2 Mobile Host Level**

This level contains the mobile replica, each replica in this level is updated frequently, and then synchronized with the master replica. Mobile host level uses the N-agent to monitor the connection with the other mobile devices through interaction with its environment via message passing. N-Agent is also used to monitor the mobility, battery, performance (or load) information, signal-to-noise ratio, read, and write counter values. The mobile host passes information to the replication manager, Context analyser with the
help of BS-Agent which detects the changes and reports it to strategy manager, the replication manager takes actions according to the changes and brings back the stability of the system. N-Agent also acts as tokenizer during update and passes the update/change list that took place during the update to the base station via message passing. N-agent and U-Agent works at the mobile host level and they are responsible for carrying the list of changes during the update and update insertion after conflict resolution respectively.

4.6 SUMMARY

Due to the dynamic nature, the integration of nomadic computing devices with the grid should be implemented in an autonomous and efficient way. To achieve this, different contextual and dynamic characteristics should be handled with agents that behave by its own. In the present chapter, an Agent based replication framework design and its components with their functions have been discussed. With its assistance, the conventional grid middleware can manage mobile devices as grid resources or resource consumers in the grid environment.