Chapter 3

A FRAMEWORK FOR MODELING AND ANALYSIS OF MOBILE AGENT BASED DISTRIBUTED NETWORK MANAGEMENT SYSTEM

3.1 INTRODUCTION

Mobile agent technology has emerged as a promising paradigm. It is much more flexible and dynamic than the Client-Server paradigm and, therefore, it has become an active research and development topic. The discussion in chapter 2 suggests that as per the research works reported in the literature, the emphasis is placed on the platform that has been used to support the specific management function(s) coupled with qualitative arguments made regarding the benefits of the use of the agent technology. However, very little work has been reported in the area of modeling and analysis of the mobile agent based network management systems.

In this chapter, a general analytical model and framework for the evaluation of various network management paradigms is being introduced [87, 92]. It is also illustrated as to how the developed analytical framework can be used to quantitatively evaluate the performances and tradeoffs in the various computing paradigms. The performances of the mobile agent based paradigm with the corresponding ones under the Client-Server mode under different scenarios have been compared.

3.2 THE VARIOUS MANAGEMENT ENTITIES AND THEIR INTERACTION MODEL

In general, the key entities involved in network management models [65, 67] are as follows:

- **Manager**. This is the entity that knows how to execute the job according to its knowledge of the management activity. It is usually housed at NMS station.
• **Agent.** This is the entity that owns and provides raw data collected from the network elements. An example is the agent in SNMP protocol that handles the Management Information Base (MIB) and provides data to the Manager.

• **Management Applications.** This is the entity that needs the result generated by the Manager. For instance, the management station has many management applications like configuration management, fault management etc. which need data provided by the manager. Generally the management task is initiated by a Manager that plays a core role in the management activity. However, the Manager should be associated with a network device that can provide sufficient computational resources for the management computation purposes.

During the management process a Manager usually needs some data stored at the individual network devices in the network. Therefore the Manager may request an Agent at each network device to send the required data. According to different management tasks, this data accessing interaction may repeat many times during the management process. The results produced thereof by the Manager are reported to some central management entity such as a management application. On the other hand the results may be used to manipulate/configure the network devices managed in the network. If the interaction happens on the same site (i.e. on the same network element), the interaction is called a local interaction. If the interaction happens between two entities associated with two different network elements, the interaction is called a remote interaction over the network. These interactions directly affect the management performance of different management paradigms in terms of increased network traffic. For example, the remote interactions between a Manager and an Agent for data accessing may generate a lot of traffic in the network and introduce significant delay as well in the management reaction time.

Consider the network performance monitoring task and its interaction sequence model, as shown in Figure 3.1. Let us assume that a management station execute a network
performance monitoring task in a certain management domain which includes nodes $N1$, $N2$ ... $Nn$.

![Management Entity and Interaction Model](image)

**Figure 3.1: Management Entity and Interaction Model.**

The data needed for the task are organized in MIBs at the Management information Bases $MIB_1$, $MIB_2$, $MIB_n$ located at nodes $N1$, $N2$ ... $Nn$ respectively (these are the Data Providers). Starting from node $N1$ up to node $Nn$, the Manager, sitting at node $N0$, iteratively accesses the data stored at each node’s MIB, as per the algorithm given in Figure 3.2.

Algorithm captures the essence of the interaction between a central manager residing on a management station and various agents residing on the remote managed devices. For any management activity, say collection of certain performance monitoring parameters, certain jobs are defined and manager runs through various jobs in succession to accomplish the management activity. It may be noted that, according to different
management jobs, this data accessing interaction between the manager and agents may repeat many times during the management process.

Algorithm dataaccess()

1. Start job J;
2. $i=1$;
3. while ($i < N_{n+1}$){
   3.1. Access MIB at node $N_i$ to get data;
   3.2. Process data to perform the relevant job;
   3.3. Repeat steps 3.1 to 3.4 till end of data from node $N_i$;
   3.4  $i++$;  // Move to next node $N_{i+1}$
}
4. Perform rest of the work;
5. close job J;

**Figure 3.2: Algorithm of Interaction Model.**

During the management process, the results generate thereof are reported by the manager to a central management entity and depending upon the site (local or remote), the performance of the deployed model is accordingly affected.

**3.3 PERFORMANCE METRICS AND PARAMETERS OF C/S AND MA MODEL**

In this work the following two performance matrices have been developed with a view to compare the performance of Client/Server and MA models of network management.

1. **Network Traffic related performance:** The following traffic related performance parameters correspond to the overhead introduced by the management application specific to a network paradigm.
   i. Traffic generated around central manager residing on the NMS,
   ii. Total management traffic generated in the network.
The above mentioned parameters can also be used to determine the potential bandwidth bottlenecks.

2. **Time related performance:** The following time related performance parameters, such as,
   i. Total time taken by typical management activities.
   ii. The remote interactions time between entities on different nodes in the network.

### 3.3.1 Analysis of Client/Server Paradigm

In Client-Server mode, a manager associated with the management station (MS) knows how to execute a given job. It also receives the result of executed job in order to ascertain the performance of each node. The MIBs at different nodes act as data providers and these nodes on which these MIBs run can also act as result receivers, only in case they need to be re-configured by the management results. The execution of the performance monitoring task can be mapped into the Client-Server paradigm as follows: the Manager at node N0 sends out a data query (with average size of $S_{req}$ bits) to the node N1.

A management agent that receives the query accesses the data from the MIB and the data is sent back to the MS. The Manager applies the management function on the data and may initiate further data queries according to the interaction model described above. Finally, when Manager finishes the computation of the data from node N1, it generates the required result. Consider Figure 3.2. Let us say $S_{req}$ is the size (bytes) of SNMP request initiated by the Manager at node N0 in Client/Server paradigm and $S_{res}$ is the response data size (bytes) accessed by the manager from node Ni (i = 1…n) which include data collected from MIB.

Considering the algorithm stated above, the average traffic around the Manager at node N0 can be computed as given below in Equation (3.1)

... (3.1)
Where: The management cost for Client/Server paradigm in terms of traffic generated around a particular (here N0) network device & p: the number of times the polling was done.

Furthermore, if \( V_{mib} \) is number of MIB variables accessed at each node, the average traffic around the Manager at node N0 can be computed as given below in Equation (3.2)

\[
\ldots (3.2)
\]

Based on Equation (3.2), the total execution time taken by a central manager in Client/Server paradigm to complete a job is given in Equation (3.3)

\[
\ldots (3.3)
\]

Where

- Total execution time (seconds) in CS paradigm.
- Bandwidth (bps) of the link between nodes N0 and Ni.
- Latency (seconds) between nodes N0 and Ni.
- Average time (seconds) for the MIB access on a given node.
- Average time (seconds) for processing of data at the central node.

Since and are average local interaction time intervals, these can be omitted while computing the average remote interaction time as given below in Equation (3.4)

\[
\ldots (3.4)
\]

Where

- Remote interaction time (seconds) in CS paradigm.
3.3.2 Performance of MA Paradigm

In mobile agent mode, as shown in Figure 3.3, a Mobile Agent capable of applying the management logic is used as the job handler. Instead of bringing the data back to the node N0 to perform the computation, the MA is sent out (with size $S_{ma}$) that contains the computation code to nodes $N1,...,Nn$ in order to access the necessary data and compute the data locally. Then, the interaction between the Manager and an Agent becomes local interaction.

![Diagram of Mobile Agent Interaction Model]

Figure 3.3: Mobile Agent Interaction Model.
On the other hand, the result generated at the node to be managed should be sent back to the management station across the network so the result reporting operation becomes a remote interaction in this case. However, if some kind of configuration is required at the node, the configuration becomes a local interaction. If the computation needs further data from other nodes, the MA will move to the next node where the data is located. Along with the MA, the partial result may also be required at the next node. If at some node $N_i$, the task is completed, then a final result is generated and the MA may be "killed".

Therefore under this scenario the average traffic around the Manager at node $N_0$ can be computed as given below in Equation (3.5)

$$\text{...(3.5)}$$

Where
- $S_{ma}$: Management cost for MA paradigm in terms of traffic generated around a particular (here $N_0$) network device,
- $S_{ma}$: The size (bytes) of the MA,
- $S_{pr}$: The size (bytes) of intermediate partial result generated in MA paradigm at each node.

The total execution time of job is as given below in Equation (3.6)

$$\text{...(3.6)}$$

Further, the average remote interaction time is as given in Equation (3.7)

$$\text{...(3.7)}$$

Where
- $T$: Total execution time (seconds) in MA paradigm.
Remote interaction time (seconds) in MA paradigm.

By removing the average local interaction time, i.e. and from the total execution time of a job, the average remote interaction time is computed.

To gain some insight into how the models and results obtained in the previous discussion can be used for different management architectures and computing paradigms under different networking management tasks and also to advantage of the mobile agent technology for network management and for the purpose of managing legacy SNMP based systems, a flexible architecture that integrates mobile agents with the SNMP protocol is being proposed here. This mobile agent based network management (MAN) framework gives the network administrator flexibilities of using the SNMP protocol or mobile agents for management according to the characteristics of the target networks and the nature of the management activities.

3.4. MOBILE AGENT BASED FRAMEWORK

The fundamental setup for network management using mobile agent have been depicted in Figure 3.4 where in network management applications based on mobile agent comprise one or more mobile agent generator nodes.

![Diagram of Mobile Agent Based Network Management]

**Figure 3.4: Framework for MA-Based Network Management**
The mobile agent for network management can collect managed devices information and delivered it to manager node or can locally perform action on managed device according to its status.

The framework consists of four major components as:
1. The Manager is responsible for launching Mobile agent service according to network management application and process returned results of mobile agents.
2. The MA migrates from a managed node to another for collecting information based on pre-defined policies.
3. The MAA is an environment for MA to receive, support the execution of MA agent at network device and dispatch MA to another managed node.
4. The MAG is an application which creates MA agent according to the requirement of network management service and with the help of MAP dispatches the created MA.

Thus using the above flexible framework, MAs could be generated, for network management, according to required network management services. To perform network management task using Mobile agent at lower i.e. device level management is performed with the help of SNMP which is one of the traditional protocol for network management. The integration of mobile agent with SNMP has been considered. Such hybrid approach has two advantages as under
1. This approach is backward compatible with existing SNMP based network management applications (craft terminal).
2. The network level disadvantages of SNMP such as scalability problem, unreliable transport protocol and security problems have been removed. The structure of such hybrid approach that combines mobile agent and SNMP for network management is shown below in Figure 3.5.

In this approach networks are managed by mobile agents using mobile agent platform at each managed device though an independent device is managed by using SNMP based management. Each MA has a specific task for network management following its own itinerary. The itinerary of MA can be static or dynamic according to network management application and network topology.
Figure 3.5: MA-Based SNMP Hybrid Model

MAG: Mobile Agent Generator
MA: Mobile agent
NE: Network element
MAA: Mobile Agent Agency
MIB: Management Information Base

Dispatch MA
Receive MA

SNMP Request & Response

ADVENTNET SNMP MEDIATION LAYER

Managed Node

CRAFT TERMINAL
3.5 COMPONENTS OF A MOBILE AGENT AGENCY

The MA based system requires mobile agent agency to support execution of mobile agent at each device on which you want to perform MA’s operations as shown in Figure 3.6.

The four major component of that agency are [70] depicted as:

**MAL**: Mobile Agent Listener is responsible for receiving mobile agent in the agency running at a managed object. MAL receive mobile agent in serialized form and de-serialized it for further processing.

**SC**: Security Component is used to check as to whether the received agent is a malicious agent or not. If the agent is malicious then its further execution is stopped.

**SFC**: Service Facility Component provides a supportive environment for agent execution. It allocates the necessary resources required by an agent to perform its assigned task.

**MFC**: Migration Facility Component provides the mobile agent migration between two agencies.

---

**Figure 3.6: General Architecture of a Mobile-Agent Agency**
These four components of mobile agent agency facilitate the fundamental services which are required for a mobile agent. There may be more than these four components in a mobile agency this depends on agency to agency.

3.6. FRAMEWORK (PROTOTYPE) IMPLEMENTATION

This section presents the details of the prototype, in terms of technologies and design algorithms used, for implementation of MA agent based network management framework. In last two decade mobile agent has become a famous technique which got huge attention from industry and research institutes, which leads to large number of mobile agent platforms developed. The implementation section is divided in three sub sections as under

1. Mobile Agent Platform – Aglet Software development kit is popular mobile agent platform. The ASDK2.0.2 is used as MAA for implementing hybrid approach for implementing NM framework.

2. SNMP Tool Kit – WebNMS Agent Toolkit Java Edition 6 (Advent Net 6.0) is used to manage low level i.e. device’s information. This information is required for network management application to analyse network status.

3. Collection of local configuration management data from managed devices using the prototype setup.

Configuration management is the network management activity which refers to setting, changing, collecting and restoring information about network devices (bridges, routers, workstations, servers, switches and others). This management activity requires to access large amount of information periodically from managed devices in order to provide statistic information and allow for capacity planning activities. The issue is that to provide this information to NM application with a lesser impact on managed network resources. The MA based NM system reduces this impact on managed network for configuration management activity by using the following steps:
3.6.1 Aglet based agent development

Step 1. Creating a new Mobile Agent.
The aglet development kit provides a method `createAglet` for creating a mobile agent i.e. called aglet. An example is given below as

```
createAglet(“ymca.edu.nmsma.performance”, “intagent.scalablemanager”, ”initarg”)
```

Step 2. Encapsulating the properties related to SNMP into Mobile Agent
To interact with SNMP agent, mobile agent must need to provide some properties of SNMP. These properties can be provided during run time or during creation time. The properties are provided using SNMP_api library. Here these properties are included at destination using SNMP_api library. For example

```
SnmpTarget target = new SnmpTarget();
target.setTargetHost("Atul-Pc");
target.setTargetPort(p);
target.setObjectID(".1.3.6.1.4.1.2.0");
target.loadMibs("C:/javaagent/WebNMS/JavaAgent/mibs/TestModule");
```

Step 3. Collecting local configuration management information
To collecting status information at managed device there is need to access one or more MIB’s objects values and then these collected values are delivered to network manager by mobile agent. Then this information is used to calculate network performance at network manager. For example

```
result = target.snmpGet();
```

Step 4. For migration of mobile agent
To migration of mobile agent can migrate from node to node using `dispatch` method. The dispatch method take argument as url of next destination i.e next managed device. For example
this.dispatch(new URL(Itinerary.next()));

The itinerary is the route of mobile agent and it can be static or dynamic according to management application.

Snmpget class code for data retrieval for managed object is an given below.

```java
//code for mobile agent for Snmpget method
import com.ibm.aglet.*;
import com.ibm.aglet.event.*;
import com.adventnet.snmp.beans.*;
import com.adventnet.snmp.snmp2.*;
public class mySnmpget extends Aglet
{
    public void run()
    {
        try {
            String result;
            SnmpTarget target = new SnmpTarget();
            // set the host in which the SNMP agent is running
            target.setTargetHost("Atul-PC");
            int p=8001;
            target.setTargetPort(p);
            // set the OID
            target.setObjectID(".1.3.6.1.4.1.2.0");
            // perform a GET request
            result = target.snmpGet();
            System.out.println("Get information is : "+result);
            this.dispose();
        }
        catch(Exception e)
        {
            System.out.println(" Exception = "+e);
        }
    }
}
```
3.7. CLIENT SERVER PARADIGM VS. MOBILE AGENT PARADIGM

In this section we compare the corresponding performances of the client server and the mobile agent paradigms from two different aspects, scalability and responsiveness, that are very critical for the effective and efficient operation of the network management architectures. Based on the Equations, (3.1) … (3.4), obtained in the previous section we conclude that the scalability of the Client-Server paradigm is mainly affected by the traffic and computational load generated around the management station (node N0). From Equation (3.1) it has been observed that the traffic around the management station is mainly generated by the data accessing from the MIBs of the nodes in the task, and it is proportional to the average number of nodes from which the data should be fetched. Also the traffic around the management station increases with the increase of the number of data accessing at the same network element. Moreover, we observe that since nearly all the computations are executed at the network management station the computational load is proportional to the number of nodes that are managed as well as the number of data accessing at a node. On the other hand, in the MA paradigm, the traffic load around the management station is mainly generated by the reporting of the results from the various network elements.

Under the assumption that the size of the MA is negligible compared to the total network management traffic, the traffic around the network management station is reduced significantly, since in general the raw data transportation from the NEs is much higher than the traffic generated by the reporting of the results from the NEs. At the same time, in the MA paradigm the computational load at the MS is reduced nearly to zero because the management task is executed locally on the NEs involved in the task. Therefore under the MA paradigm the computational load locally at the NEs may increase. However the MS is no longer a bottleneck from the point of view of both traffic and computational load. Thus overall the mobile agent paradigm presents a more scalable architecture.

The total management time of a management task can be used to measure the reaction time of the management system. Comparing the total management time of a task under
the two different paradigms using, Equation (3.4) & (3.7), we can easily see that the
difference between the two processing timing equations may increase with the increase in
the number of MIB accesses at a certain network. The reason being that in the MA
paradigm MIB accessing becomes a local interaction between the management entity and
the managed entity, whereas it is a remote interaction in the CS paradigm. The tradeoff is
that the management entity (mobile agent) should be transported among the network
elements. Once these parameters are determined for the specific task and scenario under
consideration, the difference in the remote interaction time of a management task in the
two paradigms can be obtained and evaluated quantitatively.

<table>
<thead>
<tr>
<th>Performance Matrix</th>
<th>Client/Server Model</th>
<th>Mobile Agent Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>(management cost in terms of traffic generated around a particular network device)</td>
<td>This is proportional to the number of nodes (MIB) accessed by the manager and number of times a particular node (MIB) accessed.</td>
<td>It is proportional to the size of results in bytes collected from various nodes.</td>
</tr>
<tr>
<td>(remote interaction time (seconds) in paradigm)</td>
<td>This is proportional to time taken for remote interactions done to access nodes (MIBs) and increases with the increase in number of MIBs</td>
<td>As interaction is local between the manager (MA) and managed device, it doesn’t increase with the increase in number of MIBs.</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of Client/Server V/S MA Model

Considering Equation (3.2) and (3.5), a comparison between the two paradigms is provided in Table 3.1, wherein it may be noted that given the size of the MA (Sma) is negligible as compared to the total network management traffic, the traffic (Spr)
generated around the central management station (N0) in MA based NMS is far less than that of traffic (Sreq + Sres) in Client/Server based NMS. The point to note is that Spr is only a partial report consisting of a very small amount of pre-processed data generated by the MA on the host node. On the contrary in the CS scenario a comparatively a large amount of data migration is done from the device to the central manager and the processing is done at a central point. As Spr is much less in size than (Sreq + Sres) processed by the CS model, thus, Ccs is far greater than Cma. Thus from the presented analysis it can be informed that the MA based NMS is better than Client/Server based NMS.

3.8 NUMERICAL/EXPERIMENTAL RESULTS

In the following we consider a network station that has a certain number of nodes, say 3, in its management domain. In order to perform some management task (say to detect the availability of service state on some card or port), we assume that the management station needs to check the MIB in every node in its domain, one after another, according to some certain algorithm (e.g. service state detection algorithm). After accessing the data from a MIB, algorithm is applied on the data to compute service state parameters of various facilities. The partial results are then used for the computation in the next node. In this scenario, we assume that all the nodes in the domain should be processed by the management station and the final result should be reported to the management station.

A basic setup for computing the management cost for both client/server and mobile agent based management models is shown in Figure 3.7. A centralized manager is placed on node N0 and various static SNMP agents are placed on devices/nodes Node1, Node2, Node3. The size of data transfer is calculated using “SoftPerfect Network Protocol Analyzer” sniffer tool [84]. Aglet Software development kit is (ASDK2.0.2) [51] used as Mobile agent platform for implementing NM framework. SNMP Tool Kit, WebNMS Agent Toolkit Java [38] Edition 6 (Advent Net 6.0) is used to manage low level i.e. device’s information. This information is required for network management application to analyse network status.
3.8.1 Time Related Performance

For the given experimental setup, response time for 50 get-request & its response made by a centralized manager to its static MIB agent running on each node as shown in Figure 3.8. There is a local resident daemon process running on all devices which randomly change the values of MIB variables being retrieved. An Integer32 data type stored as variables Time1, Time2 and Time3 for Node1, Node2 and Node3 is selected. All times are captured in millisecond. The various test cases generated during the experiment are given below:
Figure 3.8: Setup for C/S model for Static Agent Time Experiment

**Case 1:** The following configuration of nodes employed is as follows:

<table>
<thead>
<tr>
<th>Node0</th>
<th>Node1, Node2 and Node3</th>
</tr>
</thead>
</table>

The results obtained in terms of time = Time1, Time2 and Time3 have been tabulated in Table 3.2 and shown in Figure 3.9.
Table 3.2: Experiment result of C/S for case 1

<table>
<thead>
<tr>
<th>S.no</th>
<th>Time1</th>
<th>Time2</th>
<th>Time3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1425</td>
<td>577</td>
<td>641</td>
</tr>
<tr>
<td>2.</td>
<td>1032</td>
<td>563</td>
<td>513</td>
</tr>
<tr>
<td>3.</td>
<td>997</td>
<td>561</td>
<td>547</td>
</tr>
<tr>
<td>4.</td>
<td>907</td>
<td>500</td>
<td>561</td>
</tr>
<tr>
<td>5.</td>
<td>987</td>
<td>568</td>
<td>577</td>
</tr>
</tbody>
</table>

Figure 3.9: Timing results for case 1

Case 2: The following configuration of nodes employed is as follows:

<table>
<thead>
<tr>
<th>Node0</th>
<th>Node1, Node2 and Node3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor: Intel core 2 Duo T6500, 2.10 GHz, RAM: 4GB, Operating System: 32 bit Window 7</td>
<td>Processor: Intel core 2 Duo T6500, 2.10 GHz, RAM: 4GB, Operating System: 32 bit Window XP</td>
</tr>
</tbody>
</table>
The results obtained in terms of time = Time1, Time2 and Time3 have been tabulated in Table 3.3 and shown in Figure 3.10.

### Table 3.3: Experiment result of C/S Case 2

<table>
<thead>
<tr>
<th>S. no</th>
<th>Time1</th>
<th>Time2</th>
<th>Time3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>179</td>
<td>249</td>
<td>113</td>
</tr>
<tr>
<td>2.</td>
<td>159</td>
<td>145</td>
<td>122</td>
</tr>
<tr>
<td>3.</td>
<td>159</td>
<td>268</td>
<td>150</td>
</tr>
<tr>
<td>4.</td>
<td>239</td>
<td>131</td>
<td>210</td>
</tr>
<tr>
<td>5.</td>
<td>141</td>
<td>129</td>
<td>112</td>
</tr>
<tr>
<td>6.</td>
<td>422</td>
<td>144</td>
<td>88</td>
</tr>
<tr>
<td>7.</td>
<td>166</td>
<td>117</td>
<td>120</td>
</tr>
</tbody>
</table>

![Figure 3.10: Timing results for case 2](image)

Similarly the same sets of operations were repeated with mobile agents. The setup is shown in Figure 3.11 is used. There is a local resident daemon process running on all devices which randomly change the values of MIB variables being retrieved. An Integer32 data type stored as variables Time1, Time2 and Time3 for Node1, Node2 and Node3 is selected. All times are captured in millisecond. The various test cases generated during the experiment are given below:
Case 1: The following configuration of nodes employed is as follows:

<table>
<thead>
<tr>
<th><strong>Node0</strong></th>
<th><strong>Node1, Node2 and Node3</strong></th>
</tr>
</thead>
</table>

The results obtained in terms of time = Time1, Time2 and Time3 have been tabulated in Table 3.4 and shown in Figure 3.12.
Table 3.4: Experiment result of Mobile Agent Case 1

<table>
<thead>
<tr>
<th>S.no</th>
<th>Time1(ms)</th>
<th>Time2(ms)</th>
<th>Time3(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>5187</td>
<td>907</td>
<td>860</td>
</tr>
<tr>
<td>2.</td>
<td>5471</td>
<td>939</td>
<td>900</td>
</tr>
<tr>
<td>3.</td>
<td>5611</td>
<td>1053</td>
<td>870</td>
</tr>
<tr>
<td>4.</td>
<td>5300</td>
<td>955</td>
<td>862</td>
</tr>
<tr>
<td>5.</td>
<td>5806</td>
<td>996</td>
<td>843</td>
</tr>
</tbody>
</table>

Figure 3.12: Timing results for case 1

**Case 2:** The following configuration of nodes employed is as follows:

<table>
<thead>
<tr>
<th>Node0</th>
<th>Node1, Node2 and Node3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor: Intel core 2 Duo T6500, 2.10 GHz, RAM: 4GB, Operating System: 32 bit Window 7</td>
<td>Processor: Intel core 2 Duo T6500, 2.10 GHz, RAM: 4GB, Operating System: 32 bit Window XP</td>
</tr>
</tbody>
</table>

The results obtained in terms of time = Time1, Time2 and Time3 have been tabulated in Table 3.5 and shown in Figure 3.13.
Table 3.5 Experiment result of Mobile Agent Case 2

<table>
<thead>
<tr>
<th>S. no</th>
<th>Time1</th>
<th>Time2</th>
<th>Time3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>290</td>
<td>191</td>
<td>199</td>
</tr>
<tr>
<td>2.</td>
<td>193</td>
<td>327</td>
<td>308</td>
</tr>
<tr>
<td>3.</td>
<td>328</td>
<td>125</td>
<td>381</td>
</tr>
<tr>
<td>4.</td>
<td>315</td>
<td>582</td>
<td>310</td>
</tr>
<tr>
<td>5.</td>
<td>304</td>
<td>539</td>
<td>316</td>
</tr>
<tr>
<td>6.</td>
<td>274</td>
<td>355</td>
<td>338</td>
</tr>
<tr>
<td>7.</td>
<td>266</td>
<td>265</td>
<td>281</td>
</tr>
</tbody>
</table>

Figure 3.13: Timing results for case 2

The average value of timing results for client/server comes out to be 169 ms and for mobile agent based model it is 313 ms. The result are expected as per the mathematical computations shown earlier. For small network and low information collection client/server perform better than mobile agent but it is the other way around for large networks and more raw data exchange as is the case with SNMP based networks. This is also shown qualitatively in the next section.
3.8.2 Network Traffic Related Performance

The management cost in terms of flow of management traffic around the management station in CS paradigm for a typical node is computed as follows.

SNMP request packet size (Sreq) = 50 Bytes,
Sma (MA size) is 3 KB = 1024*3 = 3072 Bytes,
Data accessed by the task manager (Sres) in CS paradigm = α times of Sma (Size of mobile agent << raw data collected in Client/Server model),
Sr(Partial Result) in MA paradigm = 200 bytes.

\[ Ccs = (Sreq+Sres) = 50 + \alpha \times Sma \]

Putting parameters in Equation (3.1) the management cost (Ccs ) for the values of \( \alpha = 5 \) & 30 for a typical node is computed below.

**Case A: Taking \( \alpha = 5 \),**
\[
Ccs = 50 + 5 \times 3072 \\
= 50 + 15360 \\
= 15410 \text{ Bytes}
\]

**Case B: Taking \( \alpha = 30 \),**
\[
Ccs = 50 + 30 \times 3072 \\
= 50 + 92160 \\
= 92210 \text{ Bytes}
\]

Putting the parameters in Equation (3.5) the management cost in MA Paradigm for a typical node is computed below.

\[
Cma = (Sma + Sr) \\
= (3072 + 200) \\
= 3272 \text{ Bytes}
\]

The results have been tabulated in Table 3.6 for the comparison purpose.
Table 3.6 Traffic around Management Station in C/S vs MA

<table>
<thead>
<tr>
<th>No. of Nodes</th>
<th>MA</th>
<th>CSA=5</th>
<th>CSB=30</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3272</td>
<td>15410</td>
<td>92210</td>
</tr>
<tr>
<td>5</td>
<td>4072</td>
<td>77050</td>
<td>461050</td>
</tr>
<tr>
<td>10</td>
<td>5072</td>
<td>154100</td>
<td>922100</td>
</tr>
<tr>
<td>20</td>
<td>7072</td>
<td>308200</td>
<td>1844200</td>
</tr>
<tr>
<td>50</td>
<td>13072</td>
<td>770500</td>
<td>4610500</td>
</tr>
</tbody>
</table>

It may be noted from Table 3.6 that with the increase in number of nodes the management cost of C/S increases many fold as compared to MA model as also illustrated in Figure 3.14.

Figure 3.14: Traffic around management station in C/S Vs. MA
3.9 CONCLUSIONS

Mobile Agent technology has been recently used as the basis for the design and development of reliable, scalable and flexible architectures for the management of large scale distributed systems. In this chapter a generic framework that can be used for the evaluation and analysis of the performance and tradeoffs of the MA management paradigm has been presented and analyzed. Although the emphasis of this chapter is placed on the calculation of the achievable performances under the mobile agent based management strategy, for comparisons mainly purposes the corresponding performances under the CS mode are also studied and obtained for different scenarios. The developed models and framework have been used to gain some insight about the use of different management architectures and computing paradigms under different networking management tasks.

Specifically we have compared the performances of the client server paradigm and the mobile agent based architecture. The numerical results for different networking scenarios obtained thereof, demonstrate the applicability of our proposed framework, The results also quantify the corresponding tradeoffs involved, and provide some guidelines about the conditions under which the MA based approach outperforms the traditional CS approach. In-fact this analytical model could help the system designer in methodology to choose the right management paradigm for a specific management task.