4.1 INTRODUCTION

In a distributed system, the route travelled by a mobile agent is defined as its itinerary. The term itinerary is depicted as an orderly sequence of nodes that belongs to a set of nodes in the network. Generally, static itinerary with static order strategy is used when the nodes to be visited by the mobile agent are known well in advance. In this strategy, the order of the visit will not change. The frequent changes in the network traffic, communication link failures and unexpected host failures, make static travel planning impractical for real time applications. Thus, the static itinerary may not be the best approach in real network environments. Instead, the mobile agents have to be more sensitive and change their itinerary accordingly. The static itinerary with dynamic order allows the mobile agent to change the order in the visiting sequence, to overcome the network load or server failure problems.

However, here too the alternative node to be visited is included in the code before the start of the journey. Based on these limitations, dynamic itinerary with dynamic order strategy is considered as a better option for mobile agent migration. In this strategy, the next node to be visited is selected on the fly, that is, after the mobile agent completes its task in a node, and is ready for migration to the next node. The main issue in dynamic order
strategy is the criteria for node selection. To select the optimal node for
migration, it is essential to carefully select the parameters that are to be used
for the node selection. The parameter selection process should assure the user
that, at each migration the user is benefited with the maximum possible
results at any given stage. In order to provide such a migration plan, this
chapter deals with a location server based dynamic itinerary, that ensures that
the travel plan is efficient, enabling better results for information retrieval.

4.2 MOBILE AGENT ITINERARIES

Generally, the migration strategy of a mobile agent will fall into
any one of the following categories (Venkatesan et al 2011).

- Static Itinerary with Static Order (SISO)
- Static Itinerary with Dynamic Order (SIDO)
- Dynamic Itinerary with Dynamic Order (DIDO)

In SISO, the list of nodes to be visited and the order of visit are
fixed by the agent service centre well before the dispatch of a mobile agent.
Thus, it is not possible to include or exclude a node during its migration. At
the same time there may arise some situations in which a node has to be
skipped for the present, due to unexpected network traffic or link failure.
Further, a mobile agent may get blocked if it fails due to node failure. More
importantly, if a node is not available for information retrieval at the time of
the visit, the only option a mobile agent will have at that time is to skip the
node. In other words, the information available at that node is not retrieved
permanently. Thus, it is clear, that the static travel plan is not efficient for
real time applications, in which the network environment is subject to change
dynamically.
In SIDO, it is possible to change the order of visit to adapt to the dynamic changes in the network. In this approach, it is possible to skip a node that is not available at the time of the visit. The mobile agent has a chance of changing the sequence to suit the environment. At the same time, the option of inclusion or exclusion of nodes is not available. In a real time environment that supports frequent updates, a node may get updated with more information after the start of the mobile agent’s travel. As a new node can not be included in the itinerary after the start of migration, the data from that particular node is not retrieved permanently. Likewise, a node in the list may fail or the information it holds may get outdated. To handle this type of situation, SIDO is not available with adaptive options.

Basically, mobile agents are adaptive in nature. If the nodes to be visited are already fixed, the adaptive characteristic of a mobile agent is not fully utilized. For this reason, it is evident that the travel plan that better suits the mobile agent is the dynamic itinerary with dynamic order.

In the architecture presented in this thesis, mobile agents are deployed in the network by the agent service center without any predefined travel plan (DIDO). That is, the code designed for the mobile agent does not include any itinerary details. It is not necessary for an itinerary to include all the nodes of the network. The mobile agent gradually knows its travel plan along the route. After its execution at each node, the mobile agent has ‘n’ possible ways for migration (‘n’ being the number of links associated with the node).

The next node to migrate is selected once the mobile agent completes its task at the current node. Before migration, a mobile agent communicates with those ‘n’ possible nodes with a request to send the required details. The details include, the status of the node, the amount of information it holds, its processing capacity, and the number of processes in
the waiting state. After receiving the required details from those nodes, the mobile agent selects an optimal node from the received list of nodes and migrates to the selected node. This process of selection continues in all the nodes it visits. The important issue in DIDO is the criteria for optimal node selection. By the term criteria, this study means the parameters used for the selection and the order of preference to be given for these parameters.

### 4.3 Dynamic Itinerary Approaches

The next optimal node selection in dynamic itinerary is based on the parameters given by the user. At this stage, it must be noted that the order in which the parameters are given preference, is another important issue. Based on the order of preference, the dynamic itinerary can be categorized into two major types (Malakooti 2000).

- **Ordinal Preference**

  An ordinal preference is where only the ordering of the preferences is important. For example, an ordinal preference may be the one that a user prefers number of relevant documents retrieved over the computation time and prefers the computation time over the migration time.

- **Cardinal Preference**

  A cardinal preference is where the magnitude of the values matters. A cardinal preference may give a trade-off between the amount of information and the computation time and a network load trade off. For example, a user is prepared to put up with more cost in the network load, if the amount and relevance of information available at a node is exceptionally good.
4.3.1 Ordinal Preference

In the event of an information retrieval application, two approaches namely, the greedy approach and the lazy approach (Rech 2008) are discussed in this section, based on which the ordinal preference is implemented.

Whenever there is an alternative route, it will be to choose the node that provides the largest benefit for the mission. That is, if there is only one possible way of migration, then the mobile agent does not have any option in the selection of an optimal node. At the same time, if there is more than one possible route, then the greedy approach selects a node that holds a large amount of information for processing.

On the other hand, the lazy approach tries to execute the node in the fastest possible way. That is, the mobile agent selects a node, that presents the smallest computation time, ignoring the amount of information it holds, as an optimal node. By this, the lazy approach ensures that the number of nodes visited by the mobile agent within a given time is high. The motivation of the lazy approach in this regard is that, the amount of information retrieved is high if the number of nodes visited is high. However, it is not true as some nodes may not have any relevant information that matches the user’s query. In that case, even though the mobile agent visits a large number of nodes, the benefit attained may be less compared to the other approaches.

4.3.2 Cardinal Preference

In the dynamic itinerary, among the approaches that support the cardinal preference, a higher density approach (Mahalhaes 2010) that better suit this thesis is considered in this section. This approach chooses the node next to the one that presents the best relevant information per execution time rate. That is, the mobile agent selects a node that stands first in the ratio of
the amount of information retrieved per execution time. Thus the higher density approach ensures that the amount of information retrieved from a node is worth the time spent at that node.

4.3.3 Random Approach

For the benefit of comparative study, the random approach (Ota et al 2010) is also included in the dynamic migration plan strategy. The next node for migration is selected without any preferences; i.e., the mobile agent selects the node randomly. As long as the next node selection is considered, the functioning of the greedy, lazy and higher density approaches is the same. These approaches are restricted to select any one node only from the neighboring nodes. The way in which these approaches operate is shown in Figure 4.1.

Greedy / Lazy / Density Approaches

Figure 4.1 Functional Diagram of the Ordinal and Cardinal Preference-based Dynamic Itinerary
Initially, the mobile agent is dispatched to a node S1, that is identified as an optimal node by the home server, indicated as stage 1 in Figure 4.1. The mobile agent performs its assigned function at the node S1, referred to as the current node of the mobile agent. On process completion, it sends a request to the nodes S2 and S4. Figure 4.1 shows that, the mobile agent at S1 selects S4 as an optimal node and migrates to S4 (Stage 2 as shown in Figure 4.1). At this stage, the mobile agent prefers S6 over S7 and migrates to S6. Instead, if node S7 is selected, then the real problem of these approaches arises. It will become an overhead for the mobile agent to select the next node.

As a result, the mobile agent either stops migration or backtracks to its root node, as node S7 is a leaf node. If the mobile agent migrates to its root node, then the root node should contain an alternative route, through which the mobile agent continues its migration. At the same time, keeping track of the already visited nodes is a complex issue, and it actually deviates from the property of a free-roaming mobile agent. Moreover, if there is no alternative route, it is a must for a mobile agent to migrate to that node invariably, whether it holds the related data or not. In addition, it has to spend some time at that node for the selection of the optimal node. These overheads make the approaches (discussed above) inefficient, and there arises a need for a new approach that addresses these issues.

4.3.4 Location Server based Approach

The mobile agent system presented in this thesis in the event of optimal node selection, revolves around a centralized location server, as shown in Figure 4.2. In this approach, the migration of a mobile agent is based on the communication between the location server and the mobile agent. The main aim of the location server is to provide the information that is requested by a mobile agent. This sort of communication is used by the
mobile agent that aims to select an optimal node. The functioning of the location server based dynamic itinerary is shown in Figure 4.2.

Initially, the home server dispatches a mobile agent, referred to as a monitoring agent, to the location server. The function of the monitoring agent is to reside in the location server and monitor the events that are happening in the location server. The various nodes that are associated with the location server send their updates as soon as the nodes are equipped with new information. Simultaneously, the monitoring agent also updates the information it holds. By this, the monitoring agent ensures that the information it holds with regard to the nodes of the network is the latest information about those nodes. On the other hand, the home server dispatches the actual mobile agent into the network to perform the information retrieval task.

Figure 4.2  Functional Diagram of the Location Server based Dynamic Itinerary
Before migration, the mobile agent sends a ping message to the location server. The monitoring agent forwards the identity of a node to which the mobile agent needs to migrate (Step 1 in Figure 4.2) as a reply to this ping message. The monitoring agent selects this node after analyzing the node information available at the location server. A node with a higher number of relevant documents that matches the user query is the main criterion in the analysis. On receipt of a node’s identity the mobile agent migrates to that node (Step 2). This process of node selection and information retrieval continues in subsequent nodes (Steps 3, 4 and 5, 6) until the task is completed.

This location server based approach to optimal node selection as a part of the dynamic itinerary benefits the user in many ways. First, this approach confirms that the mobile agent is migrating to a node with, recently updated information; a higher amount of relevant data when compared to other nodes and is in the alive state. Second, this approach provides an opportunity to revisit a node that is not available or busy at the time of the mobile agent’s previous visit. Third, the frequency of communication required before the migration is less in the location server approach. This approach requires only a single communication between the monitoring agent and the mobile agent, in contrast to the ‘k’ communication ($k - \text{number of links}$) in the other approaches (Greedy, Lazy, Density) between the current node and the adjacent nodes. Finally, in the location server approach, the process of information retrieval and optimal node selection is done in parallel. Conversely, this is done sequentially in the other approaches.

However, as the location server is a centralized one, it is subject to single point failure. At the same time, if the mobile agents are operated in a multi-region environment, the location server is distributed as shown in the M-MASIR architecture (Figure 3.2). For a multi-region environment, it is the
responsibility of the agent monitor to provide the services of the location server. The failure of the agent monitor will not affect the system, as these types of servers are available in other regions. The failure can be detected and recovered, as these agent monitors are connected together and share information periodically.

4.4 ANALYTICAL MODEL DESCRIPTION

The distributed mobile agent system is modeled as a network of nodes ‘\(N\)’ = \(\{N_1, \ldots, N_p\}\). The mobile agent system comprises of mobile agents ‘\(a\)’ = \(\{a_1, \ldots, a_m\}\). In this context a mobile agent is located on its home server ‘\(HS(a_i)\)’ at the beginning of its task assignment. Agents that are bound to large databases, location servers in this case, are called stationary agents. All other agents are considered to be mobile. In the event of information retrieval, a series of communication steps to be processed by each agent as part of its task is given as, ‘\(C\)’ = \(\{c_i | i : 1 \ldots p\}\). Each communication step “\(c_i = <S_a, D_a, m_s, m_d>_i\)” defines that a ‘\(m_s\)’ (request message) is sent from agent ‘\(S_a\)’ (source agent) to agent ‘\(D_a\)’ (destination agent), which responds with a ‘\(m_d\)’ (reply message). The network cost for a remote communication is calculated as follows:

\[
\begin{align*}
NL_{\mathcal{A}}(c_i) = & \begin{cases} 
0 \text{ if } P(N_i) = P(N_o) \\
A_c(S_a) + N_m(m_s) + A_d(D_a) + (1 - \zeta)N_m(m_d) & \text{Otherwise}
\end{cases} 
\end{align*}
\]  

(4.1)

where, \(A_c\) – agent code size; \(N_m\) – size of the message \(m_s\) or \(m_d\); 
\(\zeta\) - compression factor

Similarly, the response time for the mobile agent is calculated as follows:

\[
RT_{MA} = \sum_{i=1}^{n} \frac{M_{i-1} + P_{i0}}{a_i}
\]

(4.2)
where, $M_t$ – Migration time between 2 nodes; $P_t$ – Processing time at each node; $n$ – Total number of nodes.

4.5 EXPERIMENTAL RESULTS AND DISCUSSION

Initially, the ascendancy of the mobile agent based information retrieval over that of the client-server based system is observed, based on the experiments conducted.

4.5.1 A Case Study of Remote Data Retrieval Using Mobile Agents

The simplest form of information retrieval is for a computer to do a linear scan through documents. However, to process large document collections spread over a number of nodes in the network, mobile agents are preferred over the traditional client-server architecture. An information retrieval application that searches for medical images and collects the required related information is implemented in this work. Consider a medical practitioner who comes across a complex report of a patient with a brain tumor complaint. As the local resources did not provide sufficient information, there arises a necessity to go in for a global search and retrieval. Generally, the medical field is characterized by three components, information, data and knowledge. These components are stored under different shapes such as sheets of paper, photos, slides and electronic files. Usually they are not available in one place at a time. Therefore, this distribution is a major problem when it is required to take a time constraint decision. It is necessary to provide ways of accessing the most relevant information in time as easily, and flexibly as possible. To meet these demands and to provide appropriate decision support, mobile agents are used.
A mobile agent is dispatched from a mobile agent launcher to perform its task. The mobile agent migrates with the scanned image of the brain tumor. At each server, the mobile agent is expected to find a relative match for the image. It is assumed that all brain tumor related images and files are available in a particular folder in remote servers. If the ratio of matching reaches the threshold value, the image and the reports related to the image are retrieved by the mobile agent. Before migration to the next server, the information is compressed locally, to reduce the overall data size of the mobile agent. The data collected from each server is appended with the existing results during the itinerary.

The performance of the mobile agent is compared with that of the traditional client-server approach to information retrieval. The various strategies involved in this comparative study are,

- **Client-Server Sequential (CSS):** The information retrieval request is sent to the identified servers sequentially. That is, a new request is sent only after the receipt of the result of the previous request.

- **Client-Server Parallel (CSP):** The request is sent to all the identified servers in parallel at a time.

- **Mobile Agent Sequential (MAS):** The service request is sent using a mobile agent to all the identified servers sequentially. In this approach, a mobile agent is sent to a new server only after the mobile agent migrates back to the sender with the results from that server.
- **Mobile Agent Parallel (MAP):** Here, multiple mobile agents are created to match the number of identified servers and all these mobile agents are dispatched one per node in parallel.

- **Mobile agent Continuous (MAC):** A mobile agent is dispatched to visit all the servers continuously. This mobile agent migrates back to the sender, only after visiting all the identified servers, with the results retrieved from each server.

### 4.5.2 Performance Analysis of the Information Retrieval Strategies

The performance metrics that have been developed for this comparative study are, (i) Time related performance and (ii) Network traffic related performance. The parameters determined to analyze the time related performance include, the migration time between two nodes, the processing time at each node, and the total turn around time of the mobile agent. The migration process consists of marshalling data and state, transmitting the code, data and state to the destination, unmarshalling the data and state and then, resuming the agent execution. Similarly, the parameters identified to analyze the network traffic related performance include, the size of the request, reply, mobile agent and transferred data, and network bandwidth.

An analytical model, derived from that of El-Gamal et al (2009), in order to compare the performance of both the mobile agents and the client-server approach is given below. This model is utilized to measure and analyze the network load and the trip time consumed by the corresponding approaches. Trip time is the total time taken by the mobile agent from its dispatch at the home server and its arrival at the home server after the task completion. Network load defines the cumulative amount of data transmitted over the network against its bandwidth.
Network Load (NL) for CS approach,

\[
NL_{CS} = n \times p \sum_{i=1}^{n} (1 - p)^{(i+1)} \times N_{RQ}^{i} + N_{NF}^{(i+1)} + N_{RP}
\]  \hspace{1cm} (4.3)

Network Load (NL) for MA approach,

\[
NL_{MA} = n \times p \sum_{i=1}^{n} (1 - p)^{(i+1)} \times A_{C}^{i} + A_{S}^{(i+1)} + N_{RQ}^{i} + (1 - \sigma)
\]  \hspace{1cm} (4.4)

Response Time (RT) for CS approach,

\[
RT_{CS} = n \times p \sum_{i=1}^{n} (1 - p)^{(i+1)} \times \{ (N_{RP} + N_{RQ}^{i} + N_{NF}^{(i+1)}) + P_{T}^{i} \}
\]  \hspace{1cm} (4.5)

Response Time (RT) for MA approach,

\[
RT_{MA} = n \times p \sum_{i=1}^{n} (1 - p)^{(i+1)} \times \{ (A_{C}^{i} + N_{RQ}^{i} + A_{S}^{(i+1)} + (1 - \sigma)N_{RP}) \times (D_{T} + M_{T}) + W_{T}^{i} + P_{T}^{i} \}
\]  \hspace{1cm} (4.6)

where, \( N_{RQ} \) – Request sent from client to server in bytes; \( N_{RP} \) – Reply received by client from server in bytes; \( N_{NF} \) – Required data not found in the server; \( \sigma \) – Data compression factor, where \( 0 \leq \sigma \leq 1 \); \( A_{C} \) – Size of mobile agent code (bytes); \( A_{S} \) – Size of mobile agent state (bytes); \( A_{D} \) – Size of data retrieved by the mobile agent from a server (bytes); \( p_{i} \) – Probability of finding data at server ‘i’, where \( 0 \leq p_{i} \leq 1 \); \( M_{T} \) – Marshalling / Unmarshalling time; \( P_{T} \) – Processing time; \( D_{T} \) – Data transfer time; \( W_{T} \) – Waiting time of request / mobile agent.

In this analysis, the servers that are requested service are pre-defined. That means, the request is made to only the servers that hold the required information. Furthermore, the mobile agent always migrates with the code, data and the state whenever it migrates between two remote servers. However, at the time of migrating from the home server to the first remote
server, it migrates with only the code and the state. Similarly, when migrating from the last server to the home server it migrates only with the data.

As a result of these considerations, the values of some of the parameters are pre-determined. $p = 1$ (required data is available in all servers); $N_{RQ} = 50$ bytes (negligible at the time of calculating network load in MB); $N_{NF} = 20$ bytes (not applicable for this analysis); $A_C = 3000$ bytes; $A_S = 0$ (weak migration is considered for this analysis); $\sigma = 0.5$ (retrieved data is compressed into half its size); $D_T = 1.2$ mbps; $W_T = 0$ (no waiting time)

The comparative study is implemented in the distributed environment connected with 15 machines. The configuration details of these machines are listed in Table 4.1. The IBM Aglet is used as the agent server to accommodate the agent. During this experiment, the migration time between the nodes, processing time at each node, the amount of data available at a node and the amount of data collected from each node is recorded. From the results attained, the details (average values), with regard to the free-roaming mobile agent are given in Table 4.2.

The response time and the network load for the various approaches discussed in this section are determined using the values shown in Table 4.2. The resultant scenarios of the performance analysis of the strategies based on the trip time based on Equation (4.2) are shown in Figure 4.3, and those on the network load based on Equation (4.1) are shown in Figure 4.4.
Table 4.1 Configurations of the Nodes used for the Experiment

<table>
<thead>
<tr>
<th>Node ID (Geographical Location)</th>
<th>IP Address</th>
<th>Processor / RAM Capacity</th>
<th>Approximate Distance (Kilometers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HS (Chennai)</td>
<td>192.168.1.10</td>
<td>2.5 GHz / 2 GB</td>
<td>-</td>
</tr>
<tr>
<td>S1 (Chennai)</td>
<td>128.57.108.112</td>
<td>1.2 GHz / 2 GB</td>
<td>&lt;1</td>
</tr>
<tr>
<td>S2 (Chennai)</td>
<td>192.168.1.8</td>
<td>1.2 GHz / 3 GB</td>
<td>4</td>
</tr>
<tr>
<td>S3 (Chennai)</td>
<td>113.112.196.245</td>
<td>2.2 GHz / 3 GB</td>
<td>32</td>
</tr>
<tr>
<td>S4 (Kavarapettai)</td>
<td>10.0.128.22</td>
<td>1.2 GHz / 2 GB</td>
<td>35</td>
</tr>
<tr>
<td>S5 (Kavarapettai)</td>
<td>113.112.196.238</td>
<td>1.2 GHz / 2 GB</td>
<td>35</td>
</tr>
<tr>
<td>S6 (Kavarapettai)</td>
<td>113.112.196.248</td>
<td>2.5 GHz / 3 GB</td>
<td>38</td>
</tr>
<tr>
<td>S7 (Bangalore)</td>
<td>210.212.247.83</td>
<td>2.2 GHz / 3 GB</td>
<td>350</td>
</tr>
<tr>
<td>S8 (Bangalore)</td>
<td>173.83.205.234</td>
<td>2.5 GHz / 3 GB</td>
<td>372</td>
</tr>
<tr>
<td>S9 (Coimbatore)</td>
<td>192.168.1.1</td>
<td>1.5 GHz / 2 GB</td>
<td>410</td>
</tr>
<tr>
<td>S10 (Coimbatore)</td>
<td>61.216.65.132</td>
<td>1.2 GHz / 2 GB</td>
<td>418</td>
</tr>
<tr>
<td>S11 (Tirunelveli)</td>
<td>110.234.80.36</td>
<td>1.5 GHz / 2 GB</td>
<td>600</td>
</tr>
<tr>
<td>S12 (Tirunelveli)</td>
<td>110.234.112.10</td>
<td>1.5 GHz / 2 GB</td>
<td>615</td>
</tr>
<tr>
<td>S13 (Trichy)</td>
<td>180.151.210.12</td>
<td>2.2 GHz / 3 GB</td>
<td>320</td>
</tr>
<tr>
<td>S14 (Trichy)</td>
<td>180.151.120.8</td>
<td>2.5 GHz / 3 GB</td>
<td>315</td>
</tr>
<tr>
<td>S15 (Trichy)</td>
<td>180.151.124.128</td>
<td>1.2 GHz / 3 GB</td>
<td>308</td>
</tr>
</tbody>
</table>
Table 4.2 Details Recorded during the Migration of the Free-roaming Mobile Agent

<table>
<thead>
<tr>
<th>Mobile Agent Migration</th>
<th>Migration Time (seconds)</th>
<th>Processing Time (seconds)</th>
<th>Available Data (MB)</th>
<th>Retrieved Data (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>To</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HS</td>
<td>S1</td>
<td>4.27</td>
<td>23.33</td>
<td>260</td>
</tr>
<tr>
<td>S1</td>
<td>S2</td>
<td>15.58</td>
<td>27.45</td>
<td>320</td>
</tr>
<tr>
<td>S2</td>
<td>S3</td>
<td>8.03</td>
<td>18.20</td>
<td>104</td>
</tr>
<tr>
<td>S3</td>
<td>S4</td>
<td>20.28</td>
<td>48.55</td>
<td>870</td>
</tr>
<tr>
<td>S4</td>
<td>S5</td>
<td>6.45</td>
<td>42.36</td>
<td>760</td>
</tr>
<tr>
<td>S5</td>
<td>HS</td>
<td>8.21</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>63.22</td>
<td>160.28</td>
<td>2314</td>
</tr>
</tbody>
</table>

HS – Home Server; S – Remote Server

Figure 4.3 Trip Time based Comparison of the Information Retrieval Strategies
From Figures 4.3 and 4.4, it is clear that the trip time of the MAC is higher than that of the parallel approaches but lesser than that of the sequential approach. Both the parallel approaches show less trip time but at the cost of heavy network load. On the other hand, the network load of the MAC is higher than that of the sequential approaches but lesser than that of the parallel approaches. Both sequential approaches show less network load but at the cost of high trip time.

More importantly, from Table 4.2, it is clear that the amount of data transfer through the network is drastically reduced in the mobile agent approach. For example, when the client-server approach transfers 2314 MB of data through the network, the mobile agent approach comfortably reduces this to a mere 37 MB of data. This shows that only 38% of the available data
resources is required by the user. This is achieved by the mobile agent (continuous) approach. From this analysis, it is evident that the free-roaming mobile agent benefits the user in both the trip time and network load. Thus, the superior performance of the free-roaming mobile agents over the other information retrieval strategies is observed.

4.5.3 Performance Analysis of the Dynamic Itinerary Patterns

The location server based dynamic itinerary is experimented with 15 nodes that are distributed geographically, and with the configurations shown in Table 4.1. The IBM Aglet is used as the agent server to accommodate the agent. It is tested for the above mentioned information retrieval application. Initially, a mobile agent is dispatched with the task of visiting all the nodes in a region. This task does not have any constraints like completion time, amount of data retrieved, etc.

Furthermore, it must be noted here that the mobile agent visits the same set of nodes for each type of travel plan. At the same time, the order or sequence of nodes to be visited will differ. For instance, the lazy approach will give preference to the nodes with less computation time, whereas the random approach selects the next node to visit, randomly. Let the order of visit differ but the task is to visit all the nodes. The amount of data available in each node and the amount of relevant data retrieved by the mobile agent is recorded as shown in Table 4.3. This table also includes the processing time at each node. The effect of the various travel plans that are tested using the details given in Tables 4.1, and 4.3, is shown in Figure 4.5. From the figure, it is evident that for any time based deadline the location server based approach shows a better result.
Table 4.3  Number and Size of Documents Available, Retrieved and the Processing Time

<table>
<thead>
<tr>
<th>Node ID</th>
<th>Amount of data available (MB)</th>
<th>Amount of relevant data retrieved (MB)</th>
<th>Number of documents available</th>
<th>Number of relevant documents retrieved</th>
<th>Processing time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>181</td>
<td>3</td>
<td>78</td>
<td>4</td>
<td>23.8</td>
</tr>
<tr>
<td>S2</td>
<td>284</td>
<td>5</td>
<td>182</td>
<td>8</td>
<td>47.6</td>
</tr>
<tr>
<td>S3</td>
<td>40</td>
<td>2</td>
<td>68</td>
<td>1</td>
<td>8.1</td>
</tr>
<tr>
<td>S4</td>
<td>260</td>
<td>12</td>
<td>132</td>
<td>18</td>
<td>45.2</td>
</tr>
<tr>
<td>S5</td>
<td>320</td>
<td>8</td>
<td>107</td>
<td>12</td>
<td>51.8</td>
</tr>
<tr>
<td>S6</td>
<td>104</td>
<td>16</td>
<td>51</td>
<td>9</td>
<td>18.4</td>
</tr>
<tr>
<td>S7</td>
<td>870</td>
<td>5</td>
<td>212</td>
<td>7</td>
<td>123.1</td>
</tr>
<tr>
<td>S8</td>
<td>760</td>
<td>10</td>
<td>187</td>
<td>21</td>
<td>104.8</td>
</tr>
<tr>
<td>S9</td>
<td>261</td>
<td>12</td>
<td>223</td>
<td>5</td>
<td>44.6</td>
</tr>
<tr>
<td>S10</td>
<td>672</td>
<td>8</td>
<td>141</td>
<td>7</td>
<td>102.7</td>
</tr>
<tr>
<td>S11</td>
<td>130</td>
<td>2</td>
<td>110</td>
<td>1</td>
<td>24.0</td>
</tr>
<tr>
<td>S12</td>
<td>227</td>
<td>14</td>
<td>142</td>
<td>12</td>
<td>28.0</td>
</tr>
<tr>
<td>S13</td>
<td>123</td>
<td>15</td>
<td>66</td>
<td>9</td>
<td>17.6</td>
</tr>
<tr>
<td>S14</td>
<td>436</td>
<td>18</td>
<td>165</td>
<td>21</td>
<td>62.4</td>
</tr>
<tr>
<td>S15</td>
<td>500</td>
<td>10</td>
<td>287</td>
<td>13</td>
<td>70.1</td>
</tr>
</tbody>
</table>

It is obvious that, for lazy approach, the time taken will always be more. However, for comparison purpose it is included in Figure 4.5. In this situation, according to the results obtained, it is clear that if the deadline for task completion is above 17 minutes, then all approaches behave equally. That means, for any travel plan, it needs a maximum of 17 minutes to visit all the nodes and collect the required data.
If the task completion time is fixed above 17 minutes, then there is no difference in all these approaches. At the same time, if the amount of data is considered for a given time, then there exists a difference between these travel plans. The differences among the approaches for a deadline of ‘n’ minutes, (i.e., the amount of data (in MB) retrieved by the end of ‘n’ minutes) is given in Table 4.4. This table also includes the total time taken (in minutes) by these approaches to complete the task. From Table 4.4, it is clear that, if the deadline for the task completion (visiting all nodes) is 20 minutes, then the amount of data retrieved by all the approaches is 140 MB and it remains constant. The exact difference between these approaches in the task completion time is given in the 6th column of Table 4.4. The problem with the greedy and lazy approaches, is the additional time they need to send a request to their neighbours and select the optimal one from the replies they receive.
Table 4.4 Cumulative Size of Data Retrieved against Task Completion (without time constraint)

<table>
<thead>
<tr>
<th>Approach</th>
<th>Cumulative amount of data retrieved (MB)</th>
<th>Task completion time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deadline (minutes)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Greedy</td>
<td>82</td>
<td>107</td>
</tr>
<tr>
<td>Lazy</td>
<td>18</td>
<td>62</td>
</tr>
<tr>
<td>Density</td>
<td>30</td>
<td>81</td>
</tr>
<tr>
<td>Random</td>
<td>53</td>
<td>83</td>
</tr>
<tr>
<td>Location Server</td>
<td>92</td>
<td>127</td>
</tr>
</tbody>
</table>

At the same time, in the location server approach, the optimal node selection process is done in parallel. When the mobile agent is performing its task in a node, the stationary agent available at the location server is determining the optimal node for migration. Another important difference is, the greedy approach looks for a node with a large amount of data, expecting that the data to be retrieved is also high. However, this is not always true. Even though a node holds a large amount of data, the actual data retrieved may be very less, and it depends solely on the nature of the request. The same thing is applicable for the lazy approach also. The random approach matches with all other approaches at some point of time. That is, a randomly selected travel plan may be the same as any approach, for example, the lazy approach, at one point. The outcome of the random approach, after every test, is not always consistent. As a result, it is not feasible to rely on the results based on the random approach. In the density approach, the ratio may be high even for a small amount of data. For example, processing 10 MB of data in 50
seconds (ratio is, 0.2), is greater than the ratio of processing 100 MB of data in 1000 seconds (ratio is, 0.1).

According to the density approach, the preference goes to 0.2, and the client can get a maximum of 50 MB of data. On the other hand, if the ratio 0.1 is selected, then there is a possibility of acquiring 100 MB of data. The point here is, for a small amount of data, the ratio may be high, and thus it may get dropped from the node selection. At the outset, the location server approach shows better results for all the deadlines. For any given dead line, the cumulative amount of data retrieved is high for the location server approach. The problem of the additional time taken for node selection found in the greedy or lazy approaches, and that of missing out a large amount of data found in the density approach is overcome, by the location server approach. The node selection process is done concurrently, and thus saves the overall trip time. Similarly, this approach makes a mobile agent to migrate only to a node that is loaded with the relevant information.

In order to strengthen the location server approach for another measure, similar test is conducted in which the task is to collect 100 MB of data. However, unlike the previous test, now the task is to retrieve 100 MB of data in quick time. In addition, the mobile agent can select any number of nodes and in any order for the task completion. Furthermore, it is not necessary to visit all the nodes of a region. Thus, the best approach is an approach that completes its task in less time. The task completion time of the various approaches with a constraint in the information retrieval task is shown in Figure 4.6.
From Figure 4.6, it is clear that in this task completion also, the location server based approach yields better results in all the cases. For example, at time 4.00 minutes, the data collected by the random is 40 MB, against the lazy in which the data size is 28 MB. However, at the end, the lazy completes its task well before the random. Similarly, at the start, the performance of the greedy is very low compared to the others (except Lazy), but eventually it is better (except location server based approach). For a higher density approach, the variation is very large between the time 2.00 minutes and 5.00 minutes. In this scenario, the size of the data retrieved is less, but the travelling time between these nodes is very high. Likewise, between the time 6.30 and 7.30 minutes the size of the data retrieved is very high, but the travelling time is much less. This type of variation makes this an inefficient approach in the information retrieval task. The achievement percentage of the task (retrieval of 100 MB data) with time constraint for the various approaches discussed in this chapter is given in Table 4.5.
Table 4.5 Cumulative size of Data Retrieved against Deadline

<table>
<thead>
<tr>
<th>For Deadline (minutes)</th>
<th>Location Server (%)</th>
<th>Greedy (%)</th>
<th>Random (%)</th>
<th>Lazy (%)</th>
<th>Density (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>67</td>
<td>52</td>
<td>38</td>
<td>14</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>88</td>
<td>80</td>
<td>56</td>
<td>42</td>
<td>56</td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>98</td>
<td>97</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

From Table 4.5, it is seen that a minimum of 15% increase is achieved in task completion by the location server approach within the deadline of 5 minutes. Similarly, for the deadline of 10 minutes, 8% increase in task completion is achieved. For information retrieval, the location server achieves it in less time when compared to the other approaches. In addition, if the task completion time is not a constraint, all the approaches behave equally. Finally, for the task completion with a deadline, once again the location server based approach performs better in all the cases.

In order to compare the location server based approach with the other approaches based on the metrics of network load and migration time, the obtained results are applied in the equations given in section 4.5.2. With regard to the data transfer principles between the remote nodes, all other approaches implements similar methodology (Figure 4.1). Hence, in the following figures, the location server based approach is compared with the greedy approach only. The resultant network load for Equations (4.3) and (4.4) and migration time for Equations (4.5) and (4.6) are shown in Figures 4.7 and 4.8 respectively. With regard to the network load, the location server based approach shows a constant load. In this approach, the
only communication between the mobile agent and the stationary agent is at the time of optimal node selection. Further, the request includes only the ping message, whereas the reply includes the identification of an optimal node. On
the other hand, for other approaches like the greedy approach, the number of communications depends on the number of links available from the current node. If there is no relevant document then no data transfer and hence the network load is not considered. However, if there is more links, the load for request and reply increases for data transfer. As the number of links may vary from each current node, the network load fluctuates for each data transfer. At the same time, with regard to the migration time, other approaches perform better than the location server approach. When the other approaches migrate to the neighbor nodes, the location server approach migrates to any one of the nodes in the network. As a result, the overall migration time for the location server is high. It must be noted here, that the processing time and optimal node selection time consumed by other approaches is not included in this migration time. Considering this aspect, when the overall trip time is considered, the location server approach performs better both with regard to the migration time and network load.

4.6 SUMMARY

- The location server based travel plan dynamically selects the next optimal node on the fly for mobile agent migration.

- The presented travel plan is compared with other travel plans that are based on cardinal preference, ordinal preference and random approach through experiments.

- The experimental results show that the number and size of the relevant documents retrieved by this travel plan are always higher than those of the other travel plans for any given deadline.
• The task of visiting all the nodes in a region within a deadline of twenty minutes is successfully completed by all the approaches.

• It is observed that, for any deadline that is less than 15 minutes, the size of the data retrieved through the location server approach is higher when compared to the other approaches.

• The time constraint mobile agent with a deadline of 5 minutes retrieves 15 MB of more data using the LS based approach than the greedy approach that comes second in data retrieval. Similarly, for a deadline of 10 minutes, 15% more efficiency is achieved over the greedy approach.

• Furthermore, the task is accomplished by the location server based dynamic itinerary in 13.32 minutes, which is 2.08 minutes earlier than the greedy approach that completes the task in 15.40 minutes.

• The given task of retrieving 100 MB of data is achieved in 7 minutes and 10 seconds by the location server based dynamic travel plan, whereas that of the greedy approach is achieved in 9 minutes, which is 1 minute and 50 seconds later.

• The performance evaluation of the mobile agent system over the traditional client-server is conducted for the identified approaches. In the event of transferring 2314 MB of data over the network, the mobile agent approach reduces it to just 37 MB of data, and thus, reduces the network load.