CHAPTER 2
LITERATURE SURVEY

2.1 INTRODUCTION

Brick-and-mortar banks may still adhere to banker’s hours — though in many cases those hours extend later in the day and into the weekend — but consumers have clearly demonstrated their desire to bank at their convenience, not the banker’s. Forrester Research, Inc. (Nasdaq: FORR) is an independent research company that provides pragmatic and forward-thinking advice to global leaders in business and technology. Forrester works with professionals in 19 key roles at major companies providing proprietary research, customer insight, consulting, events, and peer-to-peer executive
programs. For more than 27 years, Forrester has been making IT, marketing, and technology industry leaders successful every day. More than 75 percent of surveyed consumers use online banking, according to a 2008 forrester survey, and more than a quarter were increasing their usage. Consumers have embraced electronic banking, and now they are increasingly doing their electronic banking on the fly (as shown in Figure 2.1). In a poll of U.S. online banking users by CashEdge, over 32 percent said they would use mobile banking if their financial institutions offered it (The Keynote Benchmark 2010).

In the news magazine, “Mobile Marketer: The News Leader in Mobile Marketing, Media and Commerce”, the article titled “Banking is a seamless fit with mobile: TowerGroup” published in 2009 stated that the slowing economy has caused consumers to want real-time access and control of assets, resulting in greater demand for mobile banking programs. According to a study by TowerGroup, mobile banking will take over the mainstream channel by 2013, despite the recent slowdown in the banking industry. Mobile banking usage is estimated to grow from the 10 million active users of 2009 to more than 53 million active users in 2013, comprising an annual growth rate of over 50 percent.

Any bank that thinks there's no hurry to move to mobile, take note: Jibun Bank in Japan was set up as a mobile-only bank in July 2008 and has attracted 900,000 accounts in two years.

On the topic titled, “ICT SOLUTIONS FOR FINANCIAL INCLUSION – EXPERIENCE SHARING” by one of the famous nationalized bank STATE BANK OF INDIA conducted at Hyderabad on 17th February 2009, the following points are discussed towards VISION - MARCH 2012:
No. of unbanked villages to be covered: **1 lakh.** There are about 6.5 lakh villages, of which the majorities do not have access to banking services.

- Focussed shift towards smart cards, mobile banking, internet banking & kiosk banking platforms.

  Mobile devices are the most promising way to reach the masses and to create "stickiness" among current customers, due to their ability to provide services anytime, anywhere, high rate of penetration and potential to grow.

### 2.2 PRESENT SCENARIO OF MOBILE BANKING IN DEVELOPED/DEVELOPING COUNTRIES

The pioneering step of offering m-commerce was taken by NTT DOCOMO in Japan when it offered its subscribers the service of “mobile wallet” in 2004. But, the most successful example of mobile money is M-PESA, launched in Kenya by Safaricom of Kenya. It has about 7 million users out of population of 38 million & 18.3 million mobile handsets. The income of Kenyan households using M-PESA have increased by 5-30 % since they started mobile banking. Similarly, there are many more examples of success of m-commerce around the world (Sanjeev Banzal 2010).

- MTN launched Farmer’s friend, phone-based agricultural information service in Uganda
- Google & the Grameen Foundation’s “Application laboratory”, or AppLab
- China Mobile service, Nong Xin Tong, providing the same services has reached over 50 million users.
- TradeNet launched in Ghana in 2005
- CellBazar in Bangladesh
• Wizzit in South Africa
• Celpay in Zambia
• GCash & Smart money in Philippines

There are more facts that are revealed after study of success stories of various countries of m-commerce. Mobile commerce means something different to each country and to each mobile network operator. There is no universal approach to the product, its deployment and its use even within a country. The uptake for mobile commerce solutions is expected to be more aggressive in lesser developed economies in near future. Some markets are dominated by the mobile operator, while others will be dominated by the banks. Still others will have a mix. The reasons for success of m-commerce in Japan & USA are:

• Acceptability of technology
• Minimal constraints from Regulatory
• Same regulatory authority for Information Technology & telecom in Japan
• High teledensity & per capita income
• High revenue share given to the Value Added Service providers.

Reasons for success of M-PESA in Kenya are:

• The unusually high cost of money transfer
• The unusually high market share of Safaricom (80%), the main mobile operator
• The regulator’s decision to allow the scheme to proceed, even without formal regulatory approval
• The post-election violence in the country in early 2008

2.3 PRESENT SCENARIO IN INDIA

By 2012, more than four billion people are expected to own mobile phones in the world. India is the second largest nation in the world in terms of number of mobile subscribers and is growing at the fastest pace in terms of number of mobile subscribers. There are over 545 million mobile phones (as on Jan’2009) in India and about 18 million are being added every month. As per the reports available, there are about 149 million (~25%) subscribers registered for the data services (December 2009 Figure). However, as compared to number of mobile subscriber base in India, user transactions through m-commerce per day are abysmally low (5-10 million) and it can be said that presently m-commerce is in the nascent stage in India. However, it has great potential of expansion of business transaction particularly in non-cash category. If properly harnessed, it can help in facilitating inclusive growth particularly in rural India.

Financial institutions should base any decision to implement mobile banking products and services on a thorough analysis of the costs and benefits associated with such action. Some of the reasons institutions offer m-banking services include:

• Lower operating costs
• Greater geographic diversification
• Improved or sustained competitive position
• Increased customer demand for services, and
• New revenue opportunities
Mobile Banking would be enabled when the customer is able to access the financial institution's networks through a cellular phone or personal digital assistant (or similar device) via wireless networks provided by telecommunications companies. Wireless services can extend the reach and enhance the convenience of an institution’s banking products and services, provided the risks associated with the delivery channel can be managed/mitigated.

2.4 TYPES OF MOBILE BANKING

Current technology enables three primary methods to access financial data and initiate transactions from a mobile phone (as shown in Figure 2.2): Short Message service (SMS) or text messaging, mobile web, and mobile client applications (Howard Wilcox 2009). Each offers advantages and presents challenges for both consumers and bankers.

2.4.1 Short Message Service (SMS)

SMS is universal — virtually every cell phone is capable of sending and receiving text messages. It’s easy to use, affordable, requires no special software, and is carrier-independent. Form a customer’s perspective, SMS is also relatively inexpensive compared to other data services. Customers can text simple commands like BAL, DUE, or COM to have returned their

Figure 2.2 Transactions from a Mobile Phone
balance, bills due, and a list of available commands. On the downside, user experience is primitive, functionality is limited, and it is the least secure of the three channels. Among other vulnerabilities, data is stored locally on the handset and is at risk if the handset is lost or stolen. One of the major reasons that transaction based services have not taken off on SMS is because of concerns about security and because SMS doesn't enable the banks to deliver a custom user interface to make it convenient for customers to access more complex services such as transactions.

2.4.1.1 Advantages and Disadvantages

SMS has a variety of advantages and disadvantages for financial applications and services.

Advantages:

- Easy-to-use
- Common messaging tool among consumers
- Works across all wireless operators
- Affordable for consumers
- Requires no software installation
- Allows banks and financial institutions to provide real time information to customers and employees
- Stored messages can be accessed without a network connection

Disadvantages:

- Text-only and limited to 140-160 characters per message
- Does not offer a secure environment
2.4.1.2 SMS based mobile banking architecture

As shown in Figure 2.3, in order to implement an SMS service, financial institutions may choose to work with an SMS technology enabler, who will ensure that the needed connections to each SMS gateways are established in order to deliver messages reliably.

SMS technology enabler provide a set of industry standard Application Programming Interfaces (APIs) that financial institutions use to send messages to them for delivery to customers. It typically supports Hyper Text Transfer Protocol (HTTP), Short Message Peer-to-Peer (SMPP), and web services.

2.4.2 Mobile Web

The mobile web allows users to access websites from their handset. The mobile web is a channel for delivery of web content, which offers and formats content to users in awareness of the mobile context. The mobile context is characterized by the nature of personal user information needs (e.g. updating a blog, accessing travel information, receiving news update), constraints of mobile phones (i.e. screen size, keypad input) and special
capabilities (i.e. location, connection type such as 3G or Wireless Local Area Network (WLAN).

The mobile web also includes the Wireless Application Protocol (WAP), which is an open standard to enable access to the internet from a mobile device.

2.4.2.1 Advantages and disadvantages

The mobile web has a variety of advantages and disadvantages for financial applications and services.

Advantages

- User experience of browsing the internet from a mobile device is familiar and offers a rich user interface experience
- Allows end users to access corporate applications
- Secure connection can be established on most of the mobile browsers

Disadvantages

- Many non-standard variables including handsets, browsers and operating systems
- Inconsistent user experience due to varying connection speeds and handset limitations
- User needs to have a data plan, which may be a barrier to adoption among price-sensitive demographics
- No “offline” (out of the coverage) capability
2.4.3 Mobile Client Application

The third and most device-friendly access method is via a dedicated mobile application, and many financial institutions and third parties are pursuing this route to serve the rapidly growing smart phone market.

A mobile client application offers services that require rich and faster and at the same time it protects the consumer and the application on the mobile handset. If the mobile client application is installed on the handset, it can be easily upgraded whenever it is needed.

The client side application architecture requires the user to download the mobile banking software onto their phone. These Java-based systems can be very nice from a customer interaction standpoint, as a bank can offer simple, easy to use applications to provide a variety of services (Riivari 2005). The real advantage to these applications is that they can be run remotely and only need to connect to the banks systems long enough to get information and execute a transaction thereby lowering the normally high data costs that may be associated with web based applications Rajnish and Stephan (2007).

2.4.3.1 Advantages and disadvantages

Mobile client applications have number of advantages for financial institutions and customers.

Advantages

- More secured transactions than the above two types.
- It works even when there is no connection to the wireless network.
2.5 COMPARISON BETWEEN DIFFERENT TYPES OF MOBILE BANKING

A summary of the some of the important features which are useful for mobile banking, for existing three methods and the proposed Mobile Agent Banking is shown in Table 2.1.

Table 2.1 Comparison Between Different Types of Mobile Banking

<table>
<thead>
<tr>
<th>Type</th>
<th>Existing/ Proposed</th>
<th>Ubiquity</th>
<th>Ease of use User Interface</th>
<th>Affordability</th>
<th>Security</th>
<th>Rich Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS</td>
<td>Existing</td>
<td>Strong</td>
<td>Strong</td>
<td>Strong</td>
<td>Weak</td>
<td>Poor</td>
</tr>
<tr>
<td>Mobile Web</td>
<td>Existing</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Weak</td>
</tr>
<tr>
<td>Mobile client Application</td>
<td>Existing</td>
<td>Poor</td>
<td>Good</td>
<td>Moderate</td>
<td>Strong</td>
<td>Strong</td>
</tr>
<tr>
<td>Mobile Agent Banking Application</td>
<td>Proposed</td>
<td>Good</td>
<td>Good</td>
<td>Moderate</td>
<td>Strong</td>
<td>Strong</td>
</tr>
</tbody>
</table>

To summarise, mobile agent banking application is proposed as the delivery mode for mobile banking.

To summarise, there is no method available in Mobile Banking Application using Mobile Agent currently. The first phase of thesis analyzes the performance of Client server Architecture with Mobile Agent Architecture in Mobile Banking. The thesis also proposes a suitable Mobile Agent Banking Architecture to improve the performance of Mobile Banking.
2.6 MOBILE BANKING TECHNOLOGY ARCHITECTURE

2.6.1 Intra Bank Transfer

The following steps explain the transaction flow in intra bank transfer as shown in Figure 2.4:

- Bank connects to Mobile networks via Mobile Banking Application
- Sender and Receiver perform a registration at Bank
- Mobile Banking Application stores account number and Bank identification
- Sender specifies a recipient Mobile number / account number and initiates a transfer request
- Mobile Banking Application performs checks and forwards the request to the Bank
- Bank validates the From / To accounts, performs a fund transfer; responds to the sender and the receiver via Mobile Banking Application
2.6.2 Inter Bank Transfer

The following steps explain the transaction flow in inter bank transfer as shown in Figure 2.5:

- Mobile Banking Application connects to the mobile networks, and connects to a Transaction Switch, which in turn connects to Banks
- Same Sender / Receiver Registration process
- Sender (Bank 1) specifies a recipient Mobile / account number of other bank (Bank 2) and initiates a transfer request
- Bank 1 validates the From account, performs a debit and returns status to Switch
- Switch sends the credit leg to Bank 2
- Bank 2 validates the credit account, and responds to Switch
  - If invalid “To” account, sends a failure message to Switch
  - Bank 1 reverses the debit, sends out a failure message to Sender
Figure 2.5 Inter Bank Transfer
2.7 COMPARISON AMONG DIFFERENT MOBILE AGENT PLATFORMS

In this section, the review of the different mobile agent platforms has been considered.

2.7.1 Aglet

Aglet, initially developed by IBM in 1997 and maintained by the open source community since 2001, is probably the most popular mobile agent platform developed so far. Aglet is built around a single-thread model for agents and a communication infrastructure based on message passing. Both synchronous and asynchronous messages are supported. Agents in Aglet use proxies (similar to the stubs in RMI) as a convenient abstraction to refer to remote agents (e.g., to send them messages). Aglet has contributed significantly to the field of mobile agents.

2.7.2 Voyager

Voyager, developed initially by ObjectSpace in 1997 and currently by Recursion Software, is a distributed computing middleware focused towards simplifying the management of remote communications of traditional CORBA and RMI protocols. It offers facilities such as the dynamic generation of CORBA proxies, mobile code, and mobile agents. Agents communicate via traditional remote method invocations using proxies. Voyager provides location transparency through forwarding chains of proxies. Voyager is an interesting platform, with much functionality, which eases the development of distributed systems. A key disadvantage of Voyager is that it is a commercial product not available for free, which could prevent many researchers
from using it in favor of other alternatives available. The forwarding chain mechanism used to track mobile agents could also be inefficient (the whole chain must be traversed in order to locate an agent) and weak (a single broken link makes the agent unreachable).

2.7.3 Grasshopper

Grasshopper was developed by IKV++ in 1999, then became part of the commercial Enago Mobile, and today its development has probably been abandoned. It is an easy-to-use platform for mobile agents, compliant with the standards Mobile Agent System Interoperability Facility (MASIF) and Foundations of Intelligent Physical Agents (FIPA). Grasshopper system can be composed of different regions. It provides agent developers with interesting features, including a graphical user interface to manage agents, agencies, and regions. By defining regions, the developer can benefit from dynamic proxies. The main disadvantage of Grasshopper is that it is not available anymore and new versions will not appear in the future. The region server could become a bottleneck, as it must update every proxy right before using it. A disconcerting feature of Grasshopper (stated in the manual) is that a call to an agent that is moving can end up executing on the copy of the agent at origin (which will be removed once the agent arrives at its destination). Finally, as in Aglet, the same predefined method is always executed after an agent’s trip.

2.7.4 Tryllian

Tryllian, developed by the homonym company in 2001, is based on a sensing-reasoning action mechanism. It allows programmers to define a reactive (based on incoming messages) and proactive (based on heartbeats) behavior of agents. Tryllian proposes a task based
programming model and communication among agents is achieved through message passing and in accordance with the FIPA standard. It also provides a persistency service. The main disadvantage of Tryllian is that it does not offer location transparency (the current location of the target agent of a message must be known in advance). In addition, its task based and asynchronous model could be difficult to use, due to its differences with the classical procedural programming. The use of a single thread per agent could be inefficient and a limitation for the programmer. Tryllian provides a large set of configuration options, which could be overwhelming. Finally, it does not offer facilities for synchronous communication or conventional method invocation.

2.7.5 JADE

JADE, developed by Telecom Italia Lab since July 1998, and was released as open source in February 2000. It is a very popular FIPA-compliant agent platform. An agent is composed of different concurrent (and non-preemptive) behaviors, which can be added dynamically. Among the benefits, it indicates that there is a wide variety of tools provided (e.g., for remote management and monitoring of agents, and to track interchanged messages) and it can be integrated with different software such as Jess1 (a rule engine which allows JADE agents to “reason” using knowledge provided in the form of declarative rules). Finally, it is also worth mentioning its support for the development of ontologies to represent the knowledge of agents. Probably, the main disadvantage is that mobility is not a key element in JADE. Thus, it focuses on other functionalities relevant to the development of multiagent systems. The JADE built-in Agent Mobility Service supports mobility among containers within the same JADE, and researchers at the Autonomous University of Barcelona provide an
Inter-Platform Mobility Service. Proxies do not exist; instead, an agent
searches the current location of its target by querying the Agent
Management System (AMS), according to the FIPA specifications.

A summary of some features of the platforms compared is shown in
Table 2.2. The evaluation of the following features for each platform is given:

- The model of mobile agents;
- The main components in the platform;
- Whether it supports the concept of proxy or not;
- Whether the proxies continue being valid when agents move;
- Whether it supports synchronous communications and/or
  Asynchronous communications;
- Whether the agents can communicate by passing messages
  among themselves;
- Whether remote calls (RMI-like) are supported;
- Whether an agent can specify a callback method to execute on
  arrival at its destination (or, on the contrary, the same
  predefined method is always executed);
- Whether it is possible to indicate the target of a message/call by
  specifying a user-friendly name;
- Whether it allows to indicate the target of a movement by
  specifying a user-friendly name;
- Whether it is freely available for download;
- Whether it is shipped with graphical tools;
• The level of activity associated with the platform (new versions released, updates to its web page, mailing lists, etc.);

• Whether it offers security mechanisms.

In the year 2007, based on the performance analysis of mobile agent platforms have been made by Trillo et al (2007) the Voyager, the Grasshopper and the Aglet performed better than other agent systems. Out of these three mobile agent platforms, the Voyager is not available free and the Grasshopper is not anymore. Therefore the Aglet is selected as the mobile agent platform for implementing the mobile banking.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Aglet</th>
<th>Voyager</th>
<th>Grasshopper</th>
<th>Tryllian</th>
<th>JADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Events</td>
<td>Procedural</td>
<td>Procedural</td>
<td>Tasks</td>
<td>Behaviors</td>
</tr>
<tr>
<td>Elements</td>
<td>Contexts</td>
<td>Servers</td>
<td>Places</td>
<td>AFC</td>
<td>Containers</td>
</tr>
<tr>
<td></td>
<td>Aglet</td>
<td>Agents</td>
<td>Regions</td>
<td>ADK</td>
<td>Main</td>
</tr>
<tr>
<td></td>
<td>Tahiti</td>
<td></td>
<td>Agents</td>
<td>Habitats</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Agents</td>
<td>Platforms</td>
</tr>
<tr>
<td>Proxies</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Dynamic proxies</td>
<td>No</td>
<td>Yes (forwards)</td>
<td>Yes (Via Region server)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Synchronous communications</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (send and receive task)</td>
<td>No</td>
</tr>
<tr>
<td>Asynchronous communications</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Messages</td>
<td>Yes</td>
<td>No</td>
<td>Yes (FIPA)</td>
<td>Yes (FIPA)</td>
<td>Yes (FIPA)</td>
</tr>
<tr>
<td>Remote calls</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Callbacks after movements</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Call/messages by name</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes (AID)</td>
</tr>
<tr>
<td>Movements by name</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes (Configuration files)</td>
<td>Yes (Via AMS)</td>
</tr>
<tr>
<td>Available for download</td>
<td>IBM Public License</td>
<td>Not free (evaluation version)</td>
<td>Not anymore</td>
<td>LGPL</td>
<td>LGPL</td>
</tr>
<tr>
<td>GUI Tools</td>
<td>Some</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Level of activity</td>
<td>Very low</td>
<td>High</td>
<td>None</td>
<td>Medium</td>
<td>Very high</td>
</tr>
<tr>
<td>Security Mechanism</td>
<td>Basic</td>
<td>Yes (security managers, etc.)</td>
<td>Basic</td>
<td>Yes (signed agents, etc.)</td>
<td>Yes (JAAS, etc.)</td>
</tr>
</tbody>
</table>
2.8 AGLET ARCHITECTURE

Aglet is a Java based mobile agent platform and library (framework) for building mobile agents based applications. An Aglet is a Java agent which can autonomously and spontaneously move from one host to another carrying a piece of code with it. Aglet is completely written in Java, thus allowing a high portability of both the agents and the platform. Aglet includes both a complete Java mobile agent platform, with a stand-alone server called Tahiti, and a library that allows developers to build mobile agents and to embed the Aglet technology in their applications.

The Aglet architecture consists of two layers and APIs that define interfaces for accessing their functions as shown in Figure 2.6. The Aglet runtime layer is the implementation of the Aglet API, and defines the behavior of the API components, such as AgletProxy and AgletContext. It provides the fundamental functions for Aglet to be created, managed, and dispatched to remote hosts. The communication layer is primarily responsible for transferring a serialized agent to a destination and receiving it. It also supports agent-to-agent communication and facilities for agent management.
2.8.1 Aglet Runtime Layer

The Aglet runtime layer implements Aglet interfaces such as AgletContext. It also consists of a core framework and subcomponents. The core framework provides the following mechanisms fundamental to Aglet execution:

- Serialization and De-serialization of Aglet
- Class loading and transfer
- Reference management and garbage collection
The subcomponents are designed to be extensible and customizable because these services may vary depending on requirements or environments. For example, the PersistenceManager for applets may store deactivated Aglet only in the memory, or else on the Web Server if it can do so. In other cases, it may have to use the default security manager set by the Web browser.

**PersistenceManager**

The PersistenceManager is responsible for storing the serialized agent, consisting of the Aglet's code and state into a persistent medium such as a hard disk.

**CacheManager**

The CacheManager is responsible for maintaining the bytecode used by the Aglet. Because the bytecode of an incoming Aglet needs to be transferred when the Aglet moves to the next destination, the CacheManager caches all bytecode even after the corresponding class has been defined.

**SecurityManager**

The SecurityManager is responsible for protecting Aglet platforms and Aglet from malicious entities. It hooks every security-sensitive operation and checks whether the caller is permitted to perform it. There is only one instance of SecurityManager in the system, and it cannot be altered once it has been installed.

These components are defined as an interface or an abstract class, so server developers can implement these components for their own use and plug them at runtime.
The execution of Aglet is divided into the following stages as shown in Figure 2.7:

- When a running Aglet wants to migrate to a remote computer, it should send out a request to Aglet Runtime layer.
- Aglet Runtime layer will encode the Aglet's status information and code into a byte array (serialized). If it is successful, the system will pass the byte array to communication layer for processing. This layer offers interface, such as Agent Transfer Protocol (ATP), etc. ATP is a simple protocol in application layer.
- In Communication layer, system will attach related systematic information to byte stream; and migrate it to remote computer in bit stream.
- Remote computer use ATP interface that offered by Communication layer to receive byte array and system information,
then Aglet Runtime layer un-serializes the byte array and get the status information and codes of Aglet, Aglet can be executed in remote machine immediately.

![ATP Protocol](image)

**Figure 2.8 The ATP Protocol**

The Aglet uses the ATP as the default implementation of the communication layer. ATP is modeled on the HTTP protocol, and is an application-level protocol for transmission of mobile agents. To enable remote communication between agents, ATP also supports message-passing. Figure 2.8 explains the ATP Protocol.

![Aglet’s Context Environment](image)

**Figure 2.9 The Aglet’s Context Environment**
Aglet offers the basic actions in Mobile Agent context: Such as create Aglet, clone Aglet, dispatch Aglet, retract Aglet, deactivate, activate and dispose etc. The Figure 2.9 explains the Aglet’s context environment.

### 2.8.2 Aglet Life Cycle

The different states in Aglet life cycle Fragkakis et al (2007) are as follows:

- **Activated**: Aglet is loaded from storage and allowed to resume execution.
- **Deactivated**: Aglet’s execution is halted and its state is saved.
- **Cloned**: Aglet is copied for concurrent execution.
- **Disposed**: Execution of the Aglet ceases permanently.
- **Created**: Aglet is initialized for execution.
- **Dispatched**: Aglet is sent to another execution context.
- **Retracted**: Aglet is obtained from another execution context.

### 2.9 MOBILE AGENT BANKING ARCHITECTURE

![Diagram of Mobile Agent Banking Architecture](image)

*Figure 2.10 Mobile Agent Banking Architecture*
In the Figure 2.10, Mobile Browser toolkit and Mobile Agent toolkit are installed in mobile phone. The mobile user selects the required transaction using mobile browser. This information is communicated to the mobile agent platform and a mobile agent is created with the specified request. The mobile agent is transferred to the mobile agent server. The Transaction Switch manages electronic payment devices, acquires transactions across many different channels, and performs high-volume switching, routing and authorization services.

2.9.1 Benefits and Issues of Mobile Agent Banking Architecture

The benefits of Mobile Agent Banking Architecture are:

- **Reduction in network load.**
- **Overcoming network latency and**
- **Disconnected operations.**

The mobile agents do not need continuous connection with the mobile phone after it has been created. They are not affected by the sudden disconnection and the situation of turning mobile phone off for power saving. In order to get the complete benefit of mobile agent system, the system must be reliable.

For a Mobile Banking System, some of the security features that need to be provided in mobile-agent architectures are:

- **Authentication:** The mobile agent needs to have been sent from an authenticated site. A mobile agent that fails authentication can be rejected from the site or, in some cases, can be allowed to execute as an *anonymous agent* within a very restricted environment.
• **Authorization**: The mobile agent may be authorized to perform only a certain number of tasks on a system. For instance, an agent with complete authorization may be able to read, write and modify a particular resource in a system unlike an agent with limited authorization.

• **Encryption**: Encryption is also a must to ensure that the content of the mobile agent is not inspected during travel.

• **Digital signatures**: They are used to verify the claimed identity of the sender and maintain message integrity. The sender cannot later repudiate the contents of the message and the receiver cannot concoct the message itself.

The second phase of the thesis proposes a reliable Mobile Agent Banking system. The thesis also proposes a security inbuilt with the Mobile Agent Banking System to provide more secured system which is fault tolerant.

### 2.10 OBJECTIVES OF THE PROPOSED WORK

The objectives of the proposed work are as follows:

1. To analyze the performance of Client Server Architecture with Mobile Agent Banking Architecture. The analysis involves: user requests and response time.

2. To identify a reliable Mobile Agent System, a modified algorithm is proposed. The algorithm considers the following two environments: i) Mobile Phone is currently disconnected ii) Mobile Phone is moved to some other network. The proposed work suggests the solution for the above two design considerations.
3. To provide authentication in the proposed Mobile Agent Banking System, a modified Agent Based Secure Remote Password algorithm (MABSRP) is proposed. It has the advantage of being faster and not requiring the clients to store or manage long-term keys. Once the Mobile users are authenticated using MABSRP, the entire communication is encrypted using an open source encryption algorithm, AES based on session key.

4. To make the system fault tolerant, the system has to continue to provide the normal operation when the failure of one or more mobile agent is not performing the assigned task due to loss. If any agent fails, the coordinator create new agents to do the job of failed ones. To select the new coordinator, a volunteer based coordinator selection fault tolerant algorithm is proposed.

2.11 SUMMARY

In this chapter, the present scenario of mobile banking concept in developing and developed countries including India, is described. The three types of existing mobile banking namely SMS, mobile web and mobile client application are discussed and compared with the proposed Mobile Agent Banking. It is found that the Mobile Agent based mobile banking is more appropriate in all aspects.

After the type of mobile banking is selected, it is decided to implement mobile banking using mobile agents. There are number of mobile agent platforms available. To identify the suitable mobile agent platforms among Aglet, Voyager, Grossshopper, Tryllian, JADE, an analysis has been made based on various factors in the year 2007. Based on the performance analysis of mobile agent platforms made by Trillo et al (2007) among the
Voyager, the Grasshopper and the Aglet, the performance of Aglet is better than other agent systems. Out of these three mobile agent platforms, the Voyager is not available free and the Grasshopper is not available anymore. Therefore the Aglet is selected as the mobile agent platform for implementing the mobile banking.

Aglet architecture is discussed and mobile agent banking architecture is described. The benefits and issues of mobile agent banking architecture are stated clearly.