MOBILE AGENTS AND APPLICATIONS

Arjun rao, dept of CSE Anjana joshi, dept of IT, Bhagirathi hegde, Junaid shigli, dept of ECE,BVBCET
arjunrao007@gmail.com

ABSTRACT:

Mobile agents are programs that can migrate from host to host in a network, at times and to places of their own choice. The state of the running program is saved, transported to the new host, and restored, allowing the program to continue from the point of suspension. Mobile agents' ability to execute on remote hosts makes them suitable as assistants performing tasks in the network on behalf of their creators. Remote assistants operate independently of their limited network connectivity; their creators can even turn off their computers.

This paper includes a detailed study about "mobile agents". It starts with the general introduction and goes on to explain general working and properties of mobile agents. We have included two case studies on Concordia (Mitsubishi electric ITA). The case study includes the working of the systems. Then we have compared the systems working on the existing systems and systems which work on mobile agents. We have in detail discussed the application of mobile agents in e-banking and have alsospecified the general application of mobile agents. We have also addressed the challenges (mainly security) faced by the mobile agent technology. Finally we conclude the paper specifying the change that a mobile agent can bring in the society.

INTRODUCTION:

Over the years computer systems have evolved from centralized monolithic computing devices supporting static applications, into client-server environments that allow complex forms of distributed computing. Throughout this evolution limited forms of code mobility have existed- the earliest being remote job entry terminals used to submit programs to a central computer. A new phase of evolution is now under way that goes one step further, allowing complete mobility of cooperating applications among supporting platforms to form a large-scale, loosely-coupled distributed system. The catalysts for this evolutionary path are mobile software agents – programs that are goal-directed and capable of suspending their execution on one platform and moving to another platform where they resume execution. More precisely, a software agent is a program that can exercise an individual's or organization's authority, work autonomously toward a goal, and meet and interact with

other agents. Possible interactions among agents include contract and service negotiation, auctioning, and bartering. Agents may be either stationary i.e. always resident at a single platform; or mobile, capable of moving among different platforms at different times.

Prominent Properties of Mobile Agents:

Intelligence: Mobile agents employ techniques from the field of artificial intelligence.

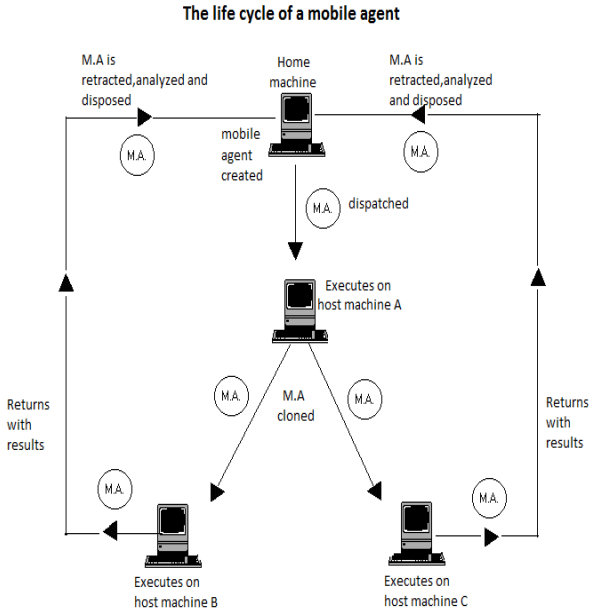
Autonomy: The agents themselves decide the sequence of actions to be performed to achieve the user's task.

Communicative Ability: They provide a user-friendly Interface.

Adaptability: Agents learn about the user's behaviour and adapt themselves to suit the user.

Working of Mobile Agents:

A mobile agent consists of the program code and the program execution state Initially a mobile agent resides on a computer called the home machine. The agent is then dispatched to execute on a remote computer called a mobile agent host /platform /server. When a mobile agent is dispatched the entire code of the mobile agent and the execution state of the mobile agent is transferred to the host. The host provides a suitable execution environment for the mobile agent to execute. The mobile agent uses resources (CPU, memory, etc.) of the host to perform its task. After completing its task on the host, the mobile agent migrates to another computer. Since the state information is also transferred to the host, mobile agents can resume the execution of the code from where they left off in the previous host instead of having to restart execution from the beginning. This continues until the mobile agent returns to its home machine after completing execution on the last machine in its itinerary.



Working of a mobile agent system:

CONCORDIA: Concordia is an infrastructure for collaborating mobile agents developed by Mitsubishi electrica USA. *Concordia* is a new framework for developing and executing highly mobile agents. *Concordia* offers a full-featured middleware infrastructure for the development and management of network-efficient mobile agent applications for accessing information anytime, anywhere, and on both wire-based and wireless devices. *Concordia* has been implemented in the Java language to ensure unimpeded interoperability and platform independence among agent applications. The design goals of *Concordia* have focused on providing complete coverage of flexible agent mobility, support for agent collaboration, persistence of agent state, reliable agent transmission, and agent security.

2. <u>System Architecture</u>: The *Concordia* infrastructure toolkit consists of a set of Java class libraries for server execution, agent application development, and agent activation. Each node in a *Concordia* system consists of a number of interacting component servers that could be executing on one or more Java virtual machines as shown in Figure.

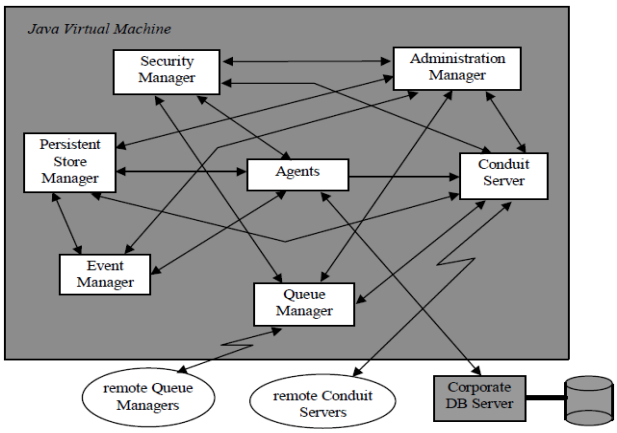


Fig. 1. Concordia System Data Flow Diagram

1. In *Concordia*, this propagation server is called the *Conduit Server*. The Conduit Server serves as the communication server for agent transfer.

2. Agent persistence is required to ensure that agents can recover successfully from system crashes. The *Persistent Store Manager* allows the internal state of agent objects to become persist able.

3. Concordia infrastructure provides support for transactional queuing of agents between Conduit Servers residing on different networks. The *Queue Manager* manages inbound and outbound queues for reliable transport of agents across a network. The Queue Manager communicates with its local Conduit Server and performs handshaking with other remote Queue Managers for reliable agent transmission.

4. Concordia's security model provides support for two types of protection:

(1)Protection of agents from being tampered with, and (2) protection of server resources from unauthorized access. *Concordia* uses the SSLv3 protocol to transmit agent information from one system to another. Agents are protected from tampering while stored on client systems during transmission and while stored on the 'persistent store'. Storage protection is handled by encryption. *Concordia* has implemented a highly flexible user-based security mechanism for server resource protection. The *Concordia Security Manager*, a Java object owned by the Java VM rather than a full-fledge process or thread, manages resource protection. Each agent is assigned an identity which allows the agent to access server resources. Resource permissions for agents can also be adjusted to alter an agent's security clearance.

5. *Concordia* system administration is handled by the '*Administration-Manager'*. The Administration Manager starts up and shuts down the other servers in the *Concordia* agent system. It also manages changes in the security profile of both agents and servers in the system and makes requests on behalf of the agent or server to the Security Manager. The Administration Manager also monitors the progress of agents throughout the network and maintains agent and system statistics.

6. Within *Concordia*, an agent's travels are described by its *Itinerary*. The Itinerary is composed of multiple *Destinations*. Each Destination describes a location to which an agent is to travel and the work the agent is to accomplish at that location. In the current implementation, location is defined by a hostname of a machine on the network and the work to accomplish is by a particular method of the agent class.

Mobility of an agent's data was accomplished using the Java Object Serialization facility. Transfer an agent state is a matter of serializing an agent's data down into a format suitable for network transmission, transmitting the data in this format, and then deserializing the data back into the original agent. Java's Object Serialization features provide an almost transparent mechanism by which Java objects can be serialized into data streams and provided suitable technology for implementing agent mobility. The *Concordia* infrastructure uses the following mechanism to support mobility of code. As an agent travels around a network its 'bytecodes' and the bytecodes of any objects it creates and stores in its member variables are loaded via a special

ClassLoader'. This ClassLoader packages theses bytecodes into a special data structure which travels with the agent. During the deserialization of the agent, the bytecodes for the agent and its related classes can be retrieved from this data structure and are used to instantiate a new copy of the agent.

Secured Authenticated Mobile Agent Based Mobile Banking System:

E-banking systems are applications that allow users to complete banking transaction. There are 4 reasons,banks choose Ebanking:

- Improve customer Service
- Reduce costs
- Inc the reactivity of the company share
- Inc market share and branding.

E-banking represents a more cost efficient channel for the banks,allowing them to charge less for transactions,and permitting the consumer to have immediate access to information related to their bankaccounts.

The major issues in E-banking system are security and authentication. The challenge is to ensure maximum mobility in mobile phone banking while at the same time providing security for users in a way that will be acceptable to both consumers and banks. A customer can send a mobile agent to perform various tasks involved in banking and get back an appropriate result

Architecture and Design of the Banking System: In this system the customer Dispatches one or more MAs,each with its list of required modes of transaction,amount and potential accounts/banks. The MA visits each bank server in turn to perform the required transaction.If the desired account is present, it processes the transaction on behalf of the user,or it moves to other bank servers.If the transaction is committed,or the bank list is exhausted,the MA returns to the customer with the details of the operations performed. 2 types of agents are implemented.They differ mainly in the different roles they can cover and/or services they can offer during trading transactions.

(a)**Bank Agent (BA)** which acts on behalf of the banking organization. It offers 2 kind of services- Account management service that includes and electronic payment service.

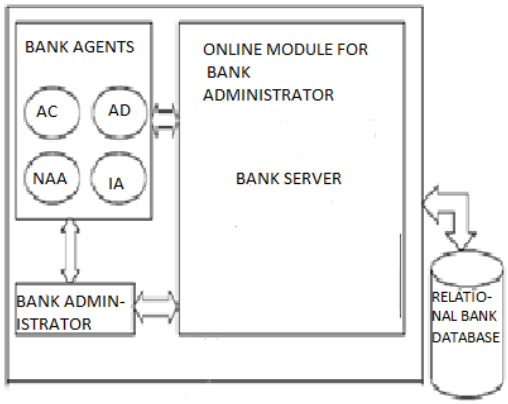
(b)**Customer Agent (CA)** which can be considered as a personal assistant that acts on the behalf of end user (customers).

Opening an account: When CA-X wants to open an account within a given bank it requests BA-Y to open an account. BA-Y will send back to CA-X the result of his demand.

Closing an account: When CA-X wants to close an account within a given bank it requests BA-Y to close its account. BA-Y

will then verify if the given account belongs to the agent CAX, and closes the account

Getting information from an account: When CA-X asks for accounts information, BA-Y receives the query and verifies the ownership of the account and the result is sent back to CA-X. The next section gives details of e-banking application.



Bank Server

This is the central module of the application. All the functionality of the bank reside here. All MAs report to this module, which performs the actual operation such as authentication and transaction on a database and returns an appropriate message to the agent. The Bank Server has four **stationary** agents called the **Authenticator Credit (AC), Authenticator Debit (AD), New Account Agent (NAA)** and **Insurance Agent (IA),** which provide second level security to the bank database after the authentication. AC is the stationary agent and helps the customers' agents to perform transactions associated with them. AD provides a similar facility to the customers' agents which want to debit their account. NAA helps in adding customers to the bank while IA supports the management of customer insurance policies.A customer MA uses the services of these bank agents to perform the required transactions by furnishing the required parameters. Customer MAs do not themselves fetch, search or update the bank database. This is done for enhancing the security of the system and prevents an authorized customer from creating a malicious agent who can manipulate his/ her account. The stationary agents perform the actual transaction and inform the customer MAs whether the transaction was successfully completed or reasons for its failure. The database is opened in read mode only.

3.4 Customer Agents (Mobile Agents)

It accepts sixteen parameters: *Login Name*, *Password*, *Agent Name*, *Agent Version*, *MethodName* (class file name of agent), *Argument List*, *Itinerary pattern type* (S: Serial, VS: Virtual Serial, P: Parallel) and *Itinerary Address* (List of Host to be visited), *Services & Location* needs to define when additional class files(other file)

are required to agent at remote site, *Agent Packing*: Mode of sealing agent, i.e., *Distributed Object* Model or nested, *Base Host*: Name of Router of the Network, where *Trusted Rerouter* is installed and *Last Field* is the location where the agent is initially submitted by the agent owner. The first two fields are for loading the authentication certificate of agent owner from the database maintained at AS and required to authenticate the agent owner on the host whose IP address/URL is given in the Itinerary Address field. This field is given to the MA, whose name is specified in the *Agent Name* field. Therefore, all the MAs have code for parsing this field to obtain the actual parameters.

A single agent is sufficient for performing the same task at different banks (single code multiple data). When the agent reaches an Agent Host(AH), it clones itself and sends these to all the places where a bank server is running. Hence all the databases, where the customer account exists, are updated simultaneously, allowing concurrency control. This feature makes handling of distributed databases possible. Following are the MAs for this application. They are:

1. **BFindAgent**: This agent is used to enquire balance in an account. It accepts three parameters,namely,*CID No.*,*Name* and *Signature* from the user.This agent has code which directly accesses the balance table of the bank database. After confirming the *CID No.* and *Signature*, it reads the amount field of the table, stores the result in a variable, and notifies it to the present AH. The host then sends this value to the user.

2. **AgentDebit**: This agent is used for debiting an account. It accepts four parameters *CID No*., *Name*, *Signature* and *Amount*. On reaching the bank server it calls the AD with these parameters. The AD then performs the actual task of debit and notifies the MA with the appropriate result about either a successful debit or reasons for failure.

3. **AgentCredit**: This agent is used for crediting an account. It accepts seven parameters: *CIDNo, Name, Signature, Amount, Cheque/DD Number, Cheque/DD Date* and *issuing bank*. After reaching the bank server it calls the AC with these parameters. The AC then performs the actual task of credit and notifies the MA with the appropriate result about either a successful credit or reasons for failing of the transaction.

4. **NewAccountAgent:** this agent takes the required info from the customers MA and creates an account for the same.

5. **QueryAgent** is a general agent, used to find the different schemes launched by banks from time-to-time. It is useful to read the general data base only.

Few more applications of Mobile Agents

E learning: Mobile agents can be distributed among different systems to acquire knowledge, this makes it a multi agent system.

Personal assistant: Based on mobile agent platform, personal commerce assistant can be developed which will handle high level shopping and other affairs on behalf of the user.

Comparative study of mobile agents and static agents:

1.Static agents perform execution on a central machine. This requires a good connection to all of the information providers. Mobile agent depends on a connection just to travel to and from the system .2. The use of static agents ensures that high priority jobs get the processor time before low priority processes. Mobile agent can determine whether or not there is enough processor time to complete a search. If not it can move to the next machine and return to the current machine later; this saves time. It can also make use of parallel processing: if it needs much of the processor power, it can distribute itself among multiple processors. 3. Static agents rely on communication protocols involving multiple interactions. The result is a lot of network traffic. Mobile agents move computation to data rather than the data to computation. This reduces network load. 4.Conventional searches send dozens or even hundreds of queries which consume a lot of bandwidth. In case of mobile agent, a large number of queries can be avoided, in the process conserving the bandwidth.

CONCERNS: Security is a major concern when using mobile agents. There is a possibility that the mobile agent returns to the home machine with malicious-ware. A virus can be disguised as a mobile agent and distributed in the network causing damage to the host machines that execute the agent. This can be avoided to some extent by introducing tracing mechanisms to records execution of mobile agent at each host. When the agent is dispatched to the next host , the trace is also sent. Using this trace, malicious actions can be detected and the malicious host can be identified. The defense mechanisms suggested try to prevent malicious actions in the first place.

If a malicious action does occur then the defense mechanisms detect it as soon as possible and take remedial action.

Conclusion:

The mobile agent technology adds a new dimension to distributed computing. Experts suggest that mobile agents will be used in many Internet applications in the years to come. However there still exist many technical hurdles that need to be tackled, the most important of them being security. Only when security issues are properly addressed, will the mobile agent technology be widely accepted.

References:

- Mobile agents: intelligent assistants on the internet
  Parineeth. M. reddy, IISC.
- A comparative study of mobile agent and client-server technologies in a real application
  R.B.Patel, K Garg
  IIT, Roorkee
- Concordia: an infrastructure for collaborating mobile agents.
  David Wong, Tom Walsh, Mike Young
  Horizon systems laboratory, mitsubshi electric
- 4. seven good reasons for mobile agents
  Danny B Lange and Mitsuru Oshima.
- 5.mobile agent based solutions for knowledge assessment in e-learning environments.
  Mihaela dinsoreanu, Claudia anghel
  Technical university of cluj-napoca, Romania