A tutorial report for SENG 609.22
Agent Based Software Engineering
Course Instructor: Dr. Behrouz H. Far

Mobile Agent: A Brief Overview

Brian Hoang

Abstract: This report provides a brief overview of mobile agents at its current state. Definition of mobile agents, their potential applications, current issues and challenges, the current state of the art and future directions will be discussed.
1. Definition of Mobile Agent

Succinctly put, mobile agents are software programs that move between computers, while autonomously trying to fulfill some specific goals given by users. Broadly speaking, agents are different from other applications in that they are goal-oriented: they represent users and act on their behalf to achieve some set goals in an autonomous manner - i.e. they control themselves, as in the decision where and when they will move to the next computer for example. Mobile agents are different from mobile processes (such as Java applets) in that agents carry both their code, their current state (i.e. data), and possibly even their current control threads so they can resume their current process at the next host. A mobile agent system consists of the agents themselves, and “docking stations” on the hosting computers that accept the agents.

2. Applications of Mobile Agent

Advocates have long proposed mobile agents for electronic commerce, software distribution, information retrieval, system administration and network management. The main benefits of mobile agents are:

2.1 Performance Benefits

Since mobile agents can move from place to place, they can improve performance by locality of preference: Running a process locally on a machine is much faster than making remote calls; therefore, it makes sense to ship the process to the machine (i.e. mobile agent.) In particular, locality of preference can greatly benefit:

- Low-resource clients. Some clients (such as PDA, cell phones, etc) do not have a lot of computing resources. It then makes sense to ship agents to machines with rich resources to perform resource-intensive routines, and then the agents return to the devices with the results only, saving precious client resources. This can greatly benefit pervasive computing, since in pervasive computing many devices are expected to be small and do not have a lot of resources.

- Disconnected clients. Some clients such as lab tops, mobile devices are not always connected to the network. Hence, it may be a good idea to ship agents to some machines to perform some works, and then when the devices reconnect to the network, they can retrieve the results from the agents.

- Low-bandwidth network. Client-server applications may require continuous messages passing. In low-bandwidth network such as cell-phone network, this can create significant strains to the network. It might be better to ship agents to some servers to perform the computations and then retrieve the results. This is particular true in information-retrieval applications. Instead of creating a permanent link between servers and clients to continuously monitor the search-engine servers for new information, which creates a lot of traffic back and fro, they can attach the agents to the servers, and when new information is available, the agents will notify the clients, saving bandwidth. Note that agents carry data
with them, so while they save continuous bandwidth, they may require large busts of bandwidth when travelling from clients to servers and back.

- Move computation closer to resources. Some servers have special resources/capabilities, and by moving the computation closer to those resources, we can improve the performance. Two examples have been mentioned in research papers. One is an example of a NASA server of 5 tera-bytes of space data. Browsing through this data remotely is clearly not as efficient as attaching an agent to the server and let the agent performs the searching locally. Another example is a specialized graphics server. Mobile agents can be used to ship the jobs to the server to perform the rendering and then send the end result back.

- Network latency. In case where there is a great network latency, traditional client-server conversations will take a long time to complete. Hence, it is desirable to save the latency by sending agents to remote servers to perform the computations and save the network trips.

- Fault tolerance. Since mobile agents do not require continuous network connections, they are not susceptible to network unavailability.

- Fault isolation. Faults at one server are isolated to the server, and do not affect a mobile agent application on a whole.

- Scalability. A mobile agent system can be easily scalable by adding additional servers.

2.2 Flexibility benefits

Some proponents of mobile agents have advocated the notion of server extensibility. The basic idea is servers will be equipped with basic operations, and the agents will contain more complicated code, and complete applications will be realized when the agents “dock” at the servers. This is also called server customization. The gained benefit is the server simplicity will help reduce maintenance (including software upgrades) costs. Along this line of argument, the following areas of mobile agent applications are proposed:

- Software management. In complex software upgrades, the upgrade might require extensive rules and computation (along with the “normal” package of new files). Such upgrades can be accomplished through mobile agents.

- Network management. In a network with many nodes, maintenance can be achieved by sending mobile agents around the network to perform the work. This is particularly useful for the Internet, for example, where the actual nodes might not be known.

2.3 A Unified Framework

At this point, some keen readers have perhaps noticed that all the potential aforementioned applications can be done without mobile agents. Mobile-agent advocates have, however, argued that mobile agents will provide a unified framework to do so, rather than each application uses its own techniques (Milojicic-2). Other proponents also argue that although individual benefits can be achieved with traditional client-server
system, mobile agents uniquely offer the aggregate benefits (i.e. the “sum” of all the individual benefits) that are difficult to match for a traditional client-server system (Harrison).

3. Issues and Challenges

3.1 Non-technical issues

- “Solution in search of a problem”. As mentioned in 2.3, all the individual benefits of mobile agents can be realized with proven existing client-server technology. It is hence difficult to push mobile agents to wide deployment in businesses. Many (not all, however) researchers have gone so far as stating that mobile agents need a “killer application” to become mainstream technology.
- Lack of applications. Related to the “solution in search of a problem”, there are not many mobile agents applications today. Mobile agents researchers tend to focus on the framework (i.e. how to make the code move, etc.) and not on the actual applications.
- Revenue and image. Current Internet service providers (e.g. Yahoo, Google, etc.) make money through advertise charges. However, agents obviously do not “watch” banner ads. Hence, there is little incentive for those service providers to adapt mobile-agent frameworks. Framework is thus needed to allow service providers and users agree on some service charges. This is not simple like e-wallet, since it is hard for the agent to decide whether the service it receives warrants the cost (e.g. the users won’t know that the search engine returns the correct matches). Another problem is business image. With mobile agent computing, the agents work behind users’ back. When a problem happens, it is hard to know what actually caused the failure. Naturally, most users will blame the service providers where it might not be their faults. For example, we often blame the web sites themselves when we cannot reach their web sites. However, this might not be their fault, but the faults of some network routers.

3.2 Technical challenges

- Security: the biggest technical challenge for mobile agent system is security. There are many components in security:
  - Authentication for both agents and hosts.
  - Authorization for both agents and hosts.
  - Denial of service attack: it is harder to prevent then denial-of-service attacks in traditional networks, since with mobile agents, the agents can be programmed to use up all the resources of the host, and it would be very hard to tell if that is an attack or if it is a legitimate computation.
  - Repudiation: since servers deal with a “representation” of users, the service providers might have problems proving that the transactions actually take place: users can deny that they have never “ordered” the agents to perform the transactions in first place.
o Copy and repay: servers can make a copy of an agent, and rerun the agents, and charge the users for multiple runs. The servers can also copy the agents and effectively “steal” them.

- Resource allocation: the servers have to limit the type of resources and/or the amount of resources that an agent is able to access (according to security credentials, and/or to the payment amount). However, once loading a process into the machine, it is not easy to control what resources the process can access or how much of the resources can be used.

- Host protection: agents can be virus, and yet they need to access resources such as database to carry out their tasks. Hence it is important to protect the servers while still allowing the agents to run (i.e. concepts such as Java “sandbox” will not work).

- Agent protection: we not only need to protect the servers, but we also need to protect the agents from being modified by the servers!

- Versioning: Distributed systems have to worry about versioning of individual components to ensure that all the components work with each other. However, there has been little research in this area with mobile agents.

- Standardizations. Many mobile-agent frameworks exist today; however, they do not interoperate with each other. Yet, in order to make mobile agents widely-deployed, it is important that the agents can work with other agents in other systems. Hence, standardization is desirable.

4. Current State of the Art

Mobile agents are still relatively new. Although there are many mobile-agent frameworks in existence today, the current state of mobile agents is still poor (Papaioannou). The problems mentioned in section 3 have still not been (fully) solved. Researches in these problem areas are still taking place (Papaioannou). In this section, we will survey some of the most well-known existing mobile-agent frameworks. Due to length restriction, only high-level overviews of each framework are provided, and details will be skipped.

- **Telescript.** Widely considered as the first mobile agent system, Telescript is developed by General Magic for Magic Cap (a handheld device, similar to the subsequent PalmPilot). Telescript is somewhat similar to Java. It is interpreted and has a built-in security model. Being the first “mobile agent”, it also introduces some initial mobile agent concepts.

- **Aglets.** Aglets is probably the most well-known mobile agent system today. It is developed by IBM, and is one of the first agent systems developed in Java. It currently enjoys a large user-base, and is currently been used in real-life E-Commerce applications.

- **Agent TCL.** Developed at Dartmouth University, Agent TCL initially starts as a Tcl/Tk based system, but has been extended to support C/C++ and Java. Agent TCL is currently used internally at Dartmouth University, as well as for information retrieval and dissemination.
**Concordia.** Developed by Mitsubishi, Concordia is a Java-based system that addresses security and reliability concerns. It uses identity, code-hashing and extension to the Java security manager to protect agents and their resources. Reliable network transmission is achieved by using a two-phase commit message queuing subsystem.

**Mole.** One of the first academic mobile agent system written in Java. It is developed by Stuttgart University, and has been used in the industry (e.g. Siemens, Tandem, Dailer-Benz). It addresses the issues of groups of agents, agent termination and security for protecting agents against malicious hosts.

**Tacoma.** A joint project between Tromso University in Norway and Cornell University, Tacoma’s main research topics include security and reliability. Tacoma addresses the operation system aspects of mobile agent systems. The researchers present services and charging, scheduling, fault tolerant support and a prototype implementation.

**Sumatra.** A Java-based mobile agent system, developed by the University of Maryland, Sumatra modifies the JVM to support the transparent migration of agents. Agents can suspend execution at any line of code, migrate and resume execution at the remote node.

**Ara.** Ara is developed at the University of Kaiserslautern. It initially uses TCL/Tk and C/C++, but now also supports Java. Like Sumatra, it also uses a modified JVM to support transparent continuation of an agent at any point of the code. Furthermore, it supports management of physical resource, such as memory.

**MOA.** MOA is recently developed by the Open Group Research Institute. It is written in Java and resembles the Telescript object model (with a few notable exceptions). The major contributions of MOA system are in the areas of resource management, transparent maintenance of communication channels across migration, and compliance with the JavaBeans component model.

**Voyager.** A widely deployed commercial system by ObjectSpace, it not only supports mobile objects and agents but also remote method invocation, object request broker, and support for DCOM. The richness of these features makes Voyager widely-used.

**MASIF.** MASIF is the first attempt to standardize mobile agent system interoperability. It has been developed by IBM, General Magic, GMD Fokus, Crystaliz and the Open Group. MASIF standardizes interoperability between mobile agent systems by specifying agent management, transfer and naming. MASIF has been accepted as an OMG technology and reference implementations are being pursued.

### 5. Future Direction

Well-known researchers in the area have different opinions on the future directions of mobile agents. However, there are certain trends that to be noted:

- Greater cooperation between mobile agent researchers and intelligent agent researchers. A lot of potential mobile agent applications depend on the agents to be intelligent (after all, representing the user requires the agents to be intelligent!)
However, mobile agent researchers focus mainly on the mobility aspect of the system (e.g. how to move code from one place to another, the security aspect of it, etc.) and do not pay much attention to the “intelligent” aspect of the system. Therefore, cooperation between the two branches will be beneficial to mobile agent frameworks.

- Some researchers have predicted a slow, gradual adoption of mobile agent technology. It is true that current mobile agent systems are not perfect; however, certain applications will need mobile agents. For instance, in space, the distances between space stations are so great that “network” latency becomes so much an issue that traditional client-server technology will not suffice. On the other hand, there are so many challenges with mobile agent technology (as noted in section 3) that it will not be widespread (at least in near future). Hence, a gradual adoption.

6. Reference