Software Technology Research Laboratory,

Mobile Agents in Distributed Environments: Principles and Paradigms

Kevin Jones
kij@dmu.ac.uk
http://www.cse.dmu.ac.uk/~kij

20th January 2005

This is part of a series of seminars offered to introduce the notion of security in Agent Distributed Systems.

This is aimed to be a general overview of the principles behind, and application of Mobile Agents.

The scope of this presentation includes:
• What are Agents (reminder)?
• What are Mobile Agents, how do they differ and present more complex security issues?
• Example Agent implementation utilising the Jade Platform.
• Agent Security Techniques.

What is an Agent? (reminder)

It would seem that there is no single definition of an agent.

The term is widely used in many spheres including:
• Distributed Computing
• Artificial Intelligence (AI)
• Network Management
• Human Computer Interaction
• Manufacturing

…..so lets start at the beginning!!

An Agent is a “Thingy”
Most literature regarding Agents introduce a definition of what an agent is in terms of its properties.

So for our purpose, an Agent possesses following properties:

- **Autonomous**
  Exercises control over its own actions
- **Goal Oriented**
  Does more than just react to the environment, has an agenda.
- **Reactive**
  Can sense and respond to changes in its environment
- **Flexible**
  Order of actions is not fixed.
- **Communicative**
  Communicates with other agents (or people)
- **Adaptive / Learning**
  Can change behaviour based on experience
- **Mobile**
  Can 'migrate' from one host to another

For our purposes, one of the more appropriate definitions to our use of Agents reads:

“An **Autonomous Agent** is a system situated within, and part of an environment, that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.”

Franklin & Graesser

(Is it an Agent, or just a program? – A Taxonomy for Autonomous Agents)

There are many 'Agent Frameworks' available to produce Agent Systems, whilst many of these are based on the 'Java' Programming language (for portability reasons) this is not always necessarily the case.

- **Ara Tcl**
  C, Tcl
- **HMain Tcl**
  Perl, Java, Tcl
- **Agents, Jade, Mole**
  Java
- **Telescript**
  Telescript
- **APRIL**
  APRIL Scripting Language

In 1996 The Foundation for Intelligent Physical Agents (FIPA) was formed to produce software standards for heterogeneous and interacting agents and agent-based systems.

This gave rise to more agent platforms, comparable to this standard:

- **APRIL**: Agent Platform (non-Java based, uses the Agent PRocess Interaction Language)
- **Comtec**: Agent Platform (requires SL2 as content language and provides minimal documentation)
- **FIPA-OS**: (only theoretical mobility – the mobility is still a prototype)
- **Grasshopper**
- **JACK**: Intelligent Agents (does not support mobility)
- **Java Agent Services (JAS)**
- **Jade / LEAP**
- **Zeus (no mobility)**

Mobile Agents have many advantages:

- **Reducing Network Load** - Reduce network interactions
- **E-Commerce** – Purchasing Agents can easily be dispatched to find, barter and buy goods.
- **Network Management & Asynchronous Execution** - Mobile devices (such as PDA's) can be turned off after an agent has migrated and the agent is still able to perform tasks within a network. It is then possible (if an 'agent-garage' is in place) for the agent to return to the device upon reconnection to the network.
- **Resource Utilisation** - Processing capabilities can be shared or transferred to devices more equipped to the task.
- **Autonomy / Intelligence** – Agents can make decisions, communicate with their environment and process information.
Example Scenarios of Mobile Agents could include:

- Searching Multiple Request from Large Database Infrastructures
- Avoidance of Network Congestion and Performance Degradation
- Shopping Agents and Booking Agents
- E-Mail Agents
- Calendar Agents
- Tracking Agents (Mobile Shadow – University of Salzburg) [http://www.mobileshadow.net/]

Mobile Agents ("Migration")
Mobile Code is not a new concept!

Even as early as the 1960’s Remote Job Entry Terminals were being used to submit programs to a central computer.

The modern implementation is Java applets, these can be considered as ‘mobile code’, downloaded via a web browser to execute on a local machine.

Mobile Agent technology, however, takes this even further, allowing complete mobility of cooperating applications among supporting platforms to form a large-scale, loosely coupled distributed system.

It is generally seen that there are TWO distinct types of migration

**Weak Migration** – is the ability of an agent to migrate its code to another host, this code may be accompanied by some initialisation data, but no migration of execution state is involved.

**Strong Migration** – Strong mobility allows an agent to transfer both the code and the execution state to another environment. It is also possible to continue execution from the point of migration.

NOTE: To implement Strong Migration requires a modified version of the Java Virtual Machine! [Jessica2]

**JADE**

JADE (Java Agent DEvelopment Framework) is a software framework to develop agent-based applications. Originally developed in 1999 it has been updated many times since and is now FIPA compliant.

JADE can be considered an agent middleware that implements an Agent Platform and a development framework.

The agent platform can be distributed on several hosts. Only one Java application, and therefore only one Java Virtual Machine (JVM), is executed on each host. Each JVM can be considered a container of agents.
Utilising the Jade Agent platform as an example:

1. Agents operate within a ‘Container’ hosted on an Agent Platform.
2. Platforms may be distributed, containing multiple containers.
3. Messages are passed, allowing Agents to communicate with other agents within a container.
4. Although it is possible to pass messages across containers (i.e. normal network load), an Agent can also migrate to a new container within the same platform and utilise local message passing, thus minimising network bandwidth usage.

Example Agent (Jade).

Behaviour Types
- “One – Shot” – Complete immediately and are executed only once.
- “Cyclic” – Never Completes, executed same operation at each invocation
- “Complex” – Actions may vary depending on the state of the agent

Jade Mobility
- “doMove()” – the doMove() command can be called to force an agent to migrate to a given environment.
- “beforeMove()” – the beforeMove() will be used to ensure that an agent is prepared to migrate.
- “afterMove()” – within the Jade framework, the execution continues from the afterMove() method once an agent has completed its migration.
Flower Shop Agent

This is an example of a flower shop agent. Although, for demonstration purposes only, it gives a good understanding of the ability of an agent. In this case to migrate, perform a given task then migrate back.

Agent Security.

Lack of Agent Security is one of the reasons why Agent technology is not being brought to the mainstream.

Both Agent Technology and thus, their associated security issues are classed as an emerging technology and in somewhat of an infancy stage.

Much research is being carried out both by academic institutions and within commercial environments to solve many of the security issues.

The Software Technology Research Laboratory (De Montfort University) is no exception. Currently the (Mobile) Agent Security Research group consists of:

Professor Hussein Zedan
Dr Antonio Cau
Kevin Jones

Much research has already been carried out by many institutions to review the issue of ‘Host Protection’.

Attacks on Hosts can include:

- Modification, deletion, hosts system files.
- Denial of Service (DoS)
- Invasion of Privacy, theft or undesirable access to data on the host system.
- Masquerading, a malicious agent claims the identity of another (possibly authorised/trusted) agent.
- Snooping, agents can monitor the behaviour of a host and gain information about its dealings.

4 Key areas of Mobile Agent Security:

- Protecting Hosts from Agents
- Protecting Hosts from Hosts
- Protecting Agents from Hosts
- Protecting Agents from Agents

Many solutions offered to solve these issues are based on serving as a deterrent, although, the combination of many or all of these techniques offers some hope for future security.

Solutions include:

- Digitally Signed Agents / Authentication, thus an audit log would provide a mechanism for compensation if a malicious or fraudulent act were undertaken. Potentially allows for agent code to be crosschecked with an MD5 hash to detect manipulations in transit.
- Sandbox / Blackbox Theory, standard Java toolkit to allow for untrusted code to be executed in an untrusted domain with privileges defined in a ‘policy’.
- State Appraisal, state is monitored and predicted to monitor for potentially harmful actions.
Protecting the Agent itself has proved to be much trickier!!

Agent V Host
A host system has control over the execution state of an agent
It controls all of the Input / Output of both the agent and itself.
All interpreted languages (such as Java based platforms – Jade) must be executed by a Virtual Machine, this VM may be modified to become malicious towards an agent.

Agent’s are vulnerable to attacks such as:

• **Access Denial** – a host may prevent the agent from executing or block the communication channels that allow message passing or migration.

• **Unauthorised access/modification of data** – the data contained within an agent (credit card information for example) may only wish to be shared with certain hosts on a multi-hop migration. This data may be vulnerable to theft or modification. [Example – booking agent, best offer]

• **Modification of code** – If an host modifies the code of an agent this could then deem the agent to be malicious.

• **Masquerade** – a host may masquerade as a different host to trick an agent into believing it is on a 'trusted' host and thus extract sensitive information.

Agent V Agent
Agents may execute in the same container (Virtual Machine) and many communicate and interact with each other.
Thus, malicious Agents may extract sensitive data from or cause harm to other agent executing within the same container.

Current Solutions:

• **Itinerary Recording**, allowing an agents path to be recorded for later inspection and repudiation. Similarly 'execution tracing' can be used.

• **Encrypted Functions**, the functions of an agent are encrypted so that the host can execute said function without deciphering the original code.

• **Obfuscated Code**, the code is scrambled, such that the original meaning is difficult to follow – this relies heavily on time for protection (i.e. the code is more difficult to follow and thus takes longer to manipulate) protecting the agent for a limited time.

• **State Appraisal**, Agents monitor their current state and monitor at which state they expect to be in next. Any abnormalities can be monitored.

Current Solutions (continued):

• **Authentication / Access Control**, Agents should authenticate themselves to perform a specific operation.
Policies can be generated to govern the access.
Formal policies can be utilised to specify security constraints from a given set of system requirements.

• **SaNTA**
SaNTa is designed to ease the development of security aware agent systems using a formal approach.
Future Work

- Develop ‘Vigilant Agents’ and thus the requirements for agent security
- Utilise the SaNTA Development Language to produce more advanced agent systems with secure vigilant agents.