MOBILITY IN AGENTS, A STUMBLING OR A BUILDING BLOCK?

John Page, Amir Padovitz and Mohamed Medhat Gaber

Centre for Distributed Systems and Software Engineering, Monash University,
900 Dandenong Rd, Caulfield East, VIC3145, Australia
Fax: + 61 3 9903 1077
{john.page, amir.padovitz, and mohamed.medhat.gaber}@infotech.monash.edu.au

Abstract

Mobility of agents has been debated over the past few years. In this paper we try to answer questions such as: Does mobility of software agents represent the future of distributed computing technology? Is mobile agents’ technology just hype? What are mobile agents’ killer applications? Are there any limitations for mobile agents? Are the security problems of mobile agents unachievable or not? These questions and others are discussed in details in this paper. Our future vision as experienced researchers in the area is also presented.

Keywords: Mobile Agents, Mobile Code, Distributed Systems.

1. Introduction

The last few years have witnessed computer networks becoming increasingly interconnected and open in nature. While this harbors good news for information exchange and data sharing, it spells out congested networks and deadlocked applications waiting for data to arrive. High demand from the hardware infrastructure to keep pace with the growing software and user requirements led to an investigation of the concept of ‘virtual connectivity’. Under this paradigm, physical networks could be used by users even when they were physically disconnected from the network, e.g. users could continue to participate in a transaction, while not being present physically. This, together with the rising popularity of web-based applications that could support mobility for users, contributed to the development of what is now known, as the ‘mobile agent’ paradigm.

In this paper we will explore characteristics of mobile agents and argue in favor and against this paradigm. This paper was inspired by the experience of authors in mobile agents research (Senousy and Gaber, 2001., Padovitz et.al, 2003., Page et.al, 2004)

While this section briefly introduces the need for the mobile agent paradigm, Section 2 examines the motivation behind the agent paradigm and summarizes the various agent definitions in vogue. Section 3 introduces the nexus of agents and mobility and how they go together. Section 4 provides various insights and arguments in favor and against mobile agents. We conclude the paper in Section 5, where we discuss our vision regarding the future of the mobile agent paradigm and how it will fit in with the evolving technology.

2. Defining Agents

Various discussions and definitions have debated the meaning of the term ‘agent’. Based upon the nature of their work, agent researchers have enumerated various characteristics of the agent paradigm. In 1996, two studies emerged that focused on summarizing the different agent definitions proposed, capturing the commonalities in the behavior of agent systems in practice. Based on their experiences in studying the agent paradigm, the researchers involved in both studies proposed ‘new’ definitions for the agent paradigm.

Bradshaw’s (1996) study first viewed an agent as an ascription and then as a description. As an ascription, he analyses the difficulties that arise in preventing researchers from coming up with a set of common criteria that can standardize the term ‘agent’. The primary difficulty in this approach is the conflict in the viewpoints of agent researchers. What might appear to be an intelligent agent to one person might be a dumb program to another. Alternatively, viewing an agent as a description allows agent researchers to attach a set of identifying criteria to the term ‘agent’. Using this criterion, the qualities of the agent can be described and their behavior can be qualitatively evaluated. This however, can also lead to conflicts regarding the actual meaning of a particular quality described as a characteristic of an agent. For example, researchers might agree that ‘autonomy’ is an essential part of an agent but might disagree on the scope of the term and its applicability. A second study attempting to come upon a formal definition was the work of Franklin and Graesser (1996). Franklin and Graesser
attempted to create a distinction between a software agent and a program. They offered a descriptive approach of agent
classification by enumerating the commonalities that existed in the different viewpoints of agent researchers. Based on these
commonalities, they enumerated a list of characteristics that agents possessed and proposed a natural taxonomy of different
agent definitions and viewpoints prevalent during that particular time. While differences in approaches existed, it was
agreed that generally, an agent can be visualized to be an entity that performs on behalf of another entity that may or may
not be a real object in the physical sense.

One of the early initiatives of the agent paradigm was the AIMA agent. AIMA was acronym for Artificial Intelligence: a
Modern Approach. Russell and Norvig (1995) defined AIMA agents as “… anything that can be viewed as perceiving its
environment through sensors and acting upon that environment through effectors”. The Brustoloni Agent described (autonomous) agents as “… systems capable of autonomous, purposeful action in the real
world”. This definition of agents focused on the need for agents to live and respond to stimuli in the real world and thus
excluded software programs in general (Franklin and Graesser, 1996).

Wooldridge and Jennings (1995) view agent as a “… a hardware or (more usually) software-based computer system…”. This was one of the early definitions that marked the transition of viewing agents as objects in the real world to abstract
tentities that existed in the software world.

Other definitions that shared the same viewpoint were Cohen’s (n.d) Sodabot Agent that embodied the idea that “Software
agents are programs that engage in dialogs [and] negotiate and coordinate transfer of information”.

Our definition of an agent finds synergy with Franklin and Graesser (1996)’s definition that attempted to summarize and
build over many of the definitions discussed here and others. Their definition viewed (autonomous) agents as, “An
autonomous agent is a system situated within and a 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”.

Thus as the agent paradigm became more refined there was a gradual transition from viewing the agent as a real world
to a virtual software entity that in most cases, was representing the authority of a real world entity.

3. Agents and Mobility

As previously discussed and further apparent from the figures of table 1, mobility has become a key component and in some
cases a need of web-based applications. The agent world was quick to see the advantages of incorporating mobility into the
agent behavior (Lange, 1999) and using the portability of Java based platform independent languages, several mobile agent
toolkits were developed.

<table>
<thead>
<tr>
<th>Region</th>
<th>Mobile Users</th>
<th>Mobile Consumers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa</td>
<td>4,900,0011</td>
<td>650,000</td>
</tr>
<tr>
<td>Asia-Pacific</td>
<td>206,500,000</td>
<td>131,750,000</td>
</tr>
<tr>
<td>Europe</td>
<td>68,850,000</td>
<td>39,350,000</td>
</tr>
<tr>
<td>Central/South America</td>
<td>18,250,000</td>
<td>282,681,500</td>
</tr>
<tr>
<td>North America</td>
<td>133,290,000</td>
<td>86790,000</td>
</tr>
<tr>
<td>Australia</td>
<td>5,250,000</td>
<td>3100,000</td>
</tr>
<tr>
<td>Former USSR</td>
<td>11,191,5008</td>
<td>191,500</td>
</tr>
<tr>
<td>World</td>
<td>448,231,500</td>
<td>282,681,500</td>
</tr>
</tbody>
</table>

Before launching into a discussion on mobile agents, it is interesting to examine the various forms of mobility that were
prevalent around the time that mobile agents arrived on the scene. This is also necessary in order to understand the
variations and the minute differences that exist between these mobile paradigms.

3.1 Mobility in Code : Historical Perspective

Four basic types of code mobility have been seen (Fischmesister, n.d). These are summarized in table 2. While COD (Code
on Demand) and REV (Remote Evaluation) proposed by Stamos and Gifford (1990), look largely similar, the difference lies
in the implementation of the mobile code. In the case of the Code on Demand paradigm, the local sends a request to the
server for the code. The code is then downloaded at the local side and executed. In the case of Remote Evaluation, the code
is pushed to the remote side for execution. After it is executed, the generated results are returned to the local side.

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Examples</th>
<th>Data (Result Sets)</th>
<th>Code Commands</th>
<th>Stack (Program Status)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Server</td>
<td>WWW, EJB, RPC, CORBA</td>
<td>Mobile</td>
<td>Static</td>
<td>Static</td>
</tr>
<tr>
<td>Code on Demand</td>
<td>Java Applets</td>
<td>Static</td>
<td>Mobile</td>
<td>Static</td>
</tr>
</tbody>
</table>
Mobile agents build on the previous forms of mobility in the sense, that all three elements of the program i.e. data, code and the program stack are mobile. In other words, the entire program can pause in mid-execution and continue its execution at another site. As obvious, mobile agent based systems can lead to the design of robust and fault tolerant systems.

While the use of the term ‘mobility’ with different forms of implementation has clouded the distinction between them, it is important to note that mobile objects, mobile code and mobile agents are three separate entities. While the Object Management Group has documented the differences that exist between agents and objects (OMG, 2000), they concede that the dividing line is blurred and it is tempting to assert that “agents are objects with attitude”, however the difficulty in implementing agents as pure objects will keep the divide standing. To summarize the differences, agents are found to be functionally smarter than objects, with communication languages of their own. Further, most agents are found to possess reactivity and autonomy (Franklin, 1996) which is another point of distinction.

Historically, General Magic came up with the first mobile agent system in 1990 by the name Telescript (White, 1995). Although it was not a commercial success, it laid the foundations for the development of other mobile agent toolkits. Around the same timeframe, various other scripting languages like TCL with its inbuilt security schema (Ousterhout, 1997) led to the development of agent systems such as AgentTCL (Gray, 1995) and Tacoma (Johansen et al.1995a). The advent of Java and the portability offered by it led to further research and Gen Magic came up again with a Java based system called Odyssey (Odyssey, 1997). Other Java based systems that came up around this time were Aglets, Grasshopper (Grasshopper, 2001), Concordia (Mitsubishi Electric. 1997) and Voyager (Objectspace, 1997) to name a few. The availability of these mobile agent toolkits led to its application in several different domain areas as described in the next section.

4 “Mobile” Agents: Building Blocks or Stumbling Blocks

While the mobility factor has seemingly opened up doors to the advancement of the agent paradigm, in the sense that it has allowed the development of several web based application it has also given voice to the skeptics, citing reasons such as safety, security and the lack of control. This section presents an objective assessment of both sides.

4.1 Building Blocks

Mobility is omnipresent. It has always been an integral part of nature and in our lives; it has manifested itself in different forms making itself an indispensable feature. This is evident from the impressive statistics gathered by Norwegian Computing Centre, stating that 25% of computer systems sold are for mobile use, Europe has 55 million cellular telephone (GSM) subscribers, USA had 2.24 million users of wireless internet-access and around 4,300 000 people are, at any time, airborne and approximately 300,000 people travel by car and train into London every morning. Technology has attempted to keep up with this increase in human mobility and the advent of mobile agents has been one of the answers. Over the last decade, mobile agents have enabled pervasive applications to adapt dynamically to changing conditions while allowing the user to maintain a high degree of mobility. It has become possible for users to remain virtually connected to a network while staying mobile. The success of this paradigm is evident from the increasing number of mobile agent toolkits that have been developed and several that are under development.

The ongoing interaction between a mobile client and a distributed application has lead to the evolution of several scenarios, which define the use of mobile agents and more importantly the requirement of mobility in agents as a critical factor. When considering distributed systems, network bandwidth is a key factor in defining the quality of service provided by an application. Restrictions imposed by slow and unreliable network connections having been overcome by mobile agents, which have made local computation possible. The disadvantages and shortcomings of other traditional methods of computing like the client and server paradigm are severely exposed when the computational requirements of data intensive applications are considered.

The mobility factor not only enables agents to migrate to several nodes, allowing it to pick and choose the best possible results but it also enables it to overcome the physical limitations of mobile clients such as low battery life of Personal Digital Assistants(PDA). EASTER (Kaneko.et.al. 2003) is one such initiative wherein the current executing application is able to migrate to another client when the battery life of the client becomes critical.

Apart from data intensive applications are some mission critical applications. Success in these applications relies on a delicate balance between its interlinked components. Space exploration is a good example of such a situation and encouraged by the high performance ratios exhibited by mobile agents, NASA is using them in designing systems for planetary extra-vehicular activity (EVA) (Clancey et.al, n.d). This will allow the development of a distributed coordination framework and support human-robotic EVA teams and will also support space exploration including potential agent manned research forays on the moon and on Mars. Supporting the need for mobility is another success story in STORMCAST based on TACOMA agents (Johansen et al.n.d b).The TACOMA agents are used to meta tag satellite images on a periodic basis.
The agent-oriented approach is particularly useful when you consider weather monitoring stations located in remote locations such as those within the Arctic circle. Obviously connectivity, bandwidth and caching capacity are all problem areas with such stations.

The STORMCAST project is a good example of a successful mobility-oriented agent approach, which exemplifies the need, and the applicability of mobility in a real-life scenario. This project is also a strong case study of a secure security infrastructure. This is made possible by injecting a high level of intelligence into the system. Users of the system can transmit their digitally signed agents to TACOMA servers for processing, thus negating the possibilities of fraudulent transactions.

An initiative of the US office of Naval Research research involving mobile agents is The Multimedia Intelligent Network of Unattended Mobile Agents project a.k.a. Minuteman (Wireless Review, 2002). This project will investigate the development of unmanned aerial vehicles (UAV), which will coordinate with unmanned ground vehicles (UGV) in simulating battle conditions.

Continuing in the same vein are the dozen or so, successful projects of the Lockheed Martin Advanced Technology Laboratories, which have been rolling out successful mobile agent based projects since 1995, including Domain Adaptive Information Systems (Dais) (McGrath, S et.al. n.d), supported by the Defense Advanced Research Projects Agency (DARPA). This project used mobile agents to query heterogeneous databases over unreliable and low-bandwidth networks. An analysis of the results displayed a significant improvement in operator decision-making.

Another successful implementation of a mobility-oriented approach is the Cooperating Agents for Specific Tasks project a.k.a. CAST. CAST uses intelligent, mobile agents to discover and analyse the location of hidden TELS by querying several distributed military data systems.

Thus, from the scenarios described above, it is clear that the requirement of a reliable computing infrastructure free from network latencies is of paramount importance. The mobility factor allows agents to migrate across and within these distributed systems and provides the required computational logic and decision-making ability in real time. The next sub-section takes a look at some of the open issues facing mobile agents.

### 4.2 Stumbling Blocks

Taking an opposing stand: most of the mobile agents’ advantages in literature can be regarded a result of mobile code rather than mobile agents: They reduce the network load; overcome network latency; encapsulate protocols; and execute asynchronously. Moreover, the notion of being an agent contradicts the notion of being mobile, since more autonomous a mobile agent is greater the size it inevitably has. In other words, features such as autonomy, robustness and reliability require more code and greater agent size; consequently damaging the quality of agility and mobility.

The time overhead for an agent to pack and unpack itself and be compiled, makes it infeasible to real-time applications. Another cost overhead is represented in the decision regarding agent migration (when, why and where to migrate) roaming the network consuming its bandwidth? An interesting point about mobile agent technology is that the mobile agent technology has been started a long time ago and during this time, we were struggling to solve the problems raised by mobile agents rather than using mobile agents to build real working systems.

Consider the following simplified cost functions for using mobile agents compared with alternative technologies.

**Cost of Remote Procedure Call (RPC)**

\[\text{Time (function execution)} + \text{[(Parameters + Results + intermediate results) / Bandwidth] + Network Overhead}\]

**Cost of Mobile Code (MC)**

\[\text{Time (MC execution)} + \text{[(MC Size + Results) / Bandwidth]}\]

**Cost of Mobile Agents (MA)**

\[\text{Time (MA execution)} + \text{Time of packing/unpacking} + \text{[(MA Code Size + MA Autonomous Code Size + Accumulated Results) / Bandwidth]}\]

Consequently, when a mobile agent has a long itinerary, it is obvious that the cost of RPC < MC < MA.

In addition, we can think of a variety of problems that specifically arise due to the combination of mobility and autonomy features of mobile agents. These become evident in distributed settings, where network and privacy considerations arise. As an example, consider the following potential weaknesses exhibited while using the mobile agent approach in a distributed e-commerce-like scenario.

- **Flooding of hosting servers** - Uncontrolled mobile agents flood the network and servers, consuming too many resources. As autonomous entities originating from different initiators, mobile agents can converge to servers of interest and consume limited resources. Their appearance in these hosts is unpredictable and uncontrollable.

- **Overheads** - A hosting server needs to be occupied with a variety of tasks, imposed by the existence of visiting agents. To name a few: management of mobile agents’ security, mobile agents’ execution and resource consumption.

- **Recovery of mobile agents** - The result of excessive resource consumption for example, would result in termination of visiting agents’ processes. This in turn, creates a critical problem of recovering these mobile agents, by the original, initiating client application. Recall that agents store information gathered during their migration, which will be lost,
once terminated by the hosting server. The latest location of the agent can be unknown to the creating application, which might not even be aware that its mobile agent code has been terminated.

- **Security** – a variety of security issues exists, covering both risks for the host and the agent. Note that agents are interpreted and executed by the hosting server and in this sense, are totally exposed to malicious attacks by hosting servers, e.g. tampering with the agent code, and are consequently not allowed to carry encryption keys. In addition, mobile agents are similar to and can be used as viruses, when injected into a hosting server. Tampering with the agent code is one of the challenges that don’t have a complete solution in the current infrastructure.

- **Distributed deadlock** – Unlike traditional deadlocks, which can be easily detected, mobile agents can be involved in distributed deadlocks that cannot be detected by the hosting server. E.g. an agent consumes a resource and migrates to another destination without releasing it. Upon arrival it awaits another resource to be released, which is held by another agent that also requires the first resource. Since these entities are autonomous and external to the hosting servers it is hard to trace and resolve such deadlocks.

- **Increase in size** - During migration, agents accumulate growing amount of information, thus becoming larger in size. This feature makes them unsuitable for operating in low bandwidth environments.

- **Contradiction to real-time needs** – Being interpreted, carrying additional code for mobility and code for autonomy, being packed and unpacked at the hosting server, mobile agents are slow and do not yield real-time results.

- **Programming complexity due to autonomy** – Although mobile agents programming abstracts some lower-level communication programming, they are inherently difficult to program when it comes to real-life multi-agent systems. The autonomy of agents requires extensive and encompassing coding that covers a variety of scenarios and possible evolution of the agent life cycle and is very hard to achieve.

In summary, it can be argued that many of the advantages suggested for mobile agents can be attributed to mobile code and that a combination of mobile code and RPC will provide in many cases better and more efficient architectural solution than using the mobile agent paradigm. The combination of mobility and autonomy introduces unnecessary problems, such as in security, trust, manageability and control. In addition, solving any mobile agent problem means more cost in terms of time, resources and size; these are costs that are not required in traditional approaches or combination of those.

5 The Future Vision

The vision of autonomous software agents, migrating between hosts and playing key roles in distributed applications, predominantly failed to materialize and evolve from research to the common market place. While this may be attributed to immaturity of mobile agent technology, with various challenges, yet to be addressed, we believe it is also the pre-maturity of the network environment, failing to exploit distinctive characteristics of this paradigm.

Evolution of distributed computer systems, characterized by rapid development of sensor technology, communication standards and ad-hoc network capabilities will critically increase the complexity of involved operations and management of tasks (Yoshimi B.). We envision these pervasive networks to be embedded with numerous computing devices, with different capabilities, limitations and goals. These systems could be characterized as highly dynamic and unpredictable environments, involving heterogeneous software platforms and technologies, servicing different organizations and individuals.

It is in this kind of environment where autonomy and survivability of independent distributed components becomes a critical feature and where we believe the paradigm of mobile agent technology becomes most valuable. Arguably, the complexity of such system weakens the ability to fully exercise centralized control and calls for autonomy of different components, facilitating different tasks for different users. Adding the unreliable nature of the underlying wireless network (e.g. frequent disconnections) and resource limitations of devices, makes agent mobility an advantage (e.g. tracking the user, utilizing the most reliable resources and being close to the source of information in face of recurrent disconnections).

While the mobile agent paradigm appears to suit these futuristic settings, it would likely fail, unless certain approaches are adopted and technological barriers are overcome. We believe that the following suggestions and issues need to be addressed to allow successful deployment of mobile agent technology in distributed applications.

- Mobile agents’ itineraries should be as short as possible, minimizing security problems and preserving bandwidth consumption that arise with lengthy itineraries.

- Mobile agents’ functionality should be as simple as possible, thereby easing code verification for security purposes and preserving limited bandwidth, especially in wireless environments.

- Mobile agents’ intermediate results should be limited in size, avoiding bandwidth consumption, especially in ubiquitous environments, and reducing security problems that arise when personal data is carried by the agent.

- Mobile agents’ systems should be combined with other distributed computing technologies in order to maximize the benefit of hybrid approaches. Distributed data mining using a hybrid approach is a typical example. Mobile agents should be combined with cost estimators that minimize the cost of a mobile agent process with the combination of other technologies.

- There is a real need for a mobile agent benchmark that could be used for mobile agent testing and debugging, as well as for comparing performance with other technologies.
Over the last fifteen years, while a lot of effort and funding has gone into the evolution of the mobile agent paradigm, it is still far from being accepted as the common vehicle for mobility based transactions. A realization of the above mentioned possibilities might help the paradigm in achieving the respect and support of the pervasive world. To make this dream a reality it is up to researchers to identify challenge problems in various aspects of the paradigm and to focus on those challenging problems.

References
Clancey, B et.al, n.d. Distributed Human-Robotic EVA System for Surface Operation.http://is.arc.nasa.gov/HCC/tasks/MblAgt.html Date of Access 12/10/2003
Johansen et al.1995a, Johansen,D.,Renesse,R.V.& Schneider,F.B., An Introduction to the TACOMA Distributed System Technical Report Department of Computer Science, University of Tromso.