

1 Introduction

The term "agent" is frequently heard today. While it means a variety of things to a variety of people, it is commonly defined as an independent software program which runs on behalf of a network user. An agent may run when the user is disconnected from the network, even involuntarily. This chapter contains agent's concepts, characteristics and architecture besides to multi-agent systems and the most popular applications using the agent technology. It also gives more details about the mobile agent: its definition, application and characteristics.

2 Agents

2.1 ABOUT AGENTS

Agents are considered one of the most important paradigms that on the one hand may improve current methods for conceptualizing, designing and implementing software systems, and that may provide the solution to the legacy software integration problem, on the other hand. [5]

2.1.1 Definition

The term "agent", or software agent, has found its way into a number of technologies and has been widely used, for example, in artificial intelligence, databases, operating systems and computer networks literature. Although there is no single definition of an agent (see, for example, Genesereth and Ketchpel, 1994; Wooldridge and Jennings, 1995; Russell and Norvig, 2003), all definitions agree that an agent is essentially a special software component that has autonomy that provides an interoperable interface to an arbitrary system and/or behaves like a human agent, working for some clients in pursuit of its own agenda. Even if an agent system can be based on a solitary agent working within an environment and if necessary interacting with its users, usually it consists of multiple agents. These multi-agent systems (MAS) can model complex systems and introduce the possibility of agents having common or conflicting goals. These agents may interact with each other both indirectly (by acting on the environment) or directly (via communication and negotiation). Agents may decide to cooperate for mutual benefit or may compete to serve their own interests. Therefore, an agent is autonomous, because it operates without the direct intervention of humans and has control over its actions and internal state. [5]

2.2 Classification of Software Agents

According to Gilbert et al. [20], software agents can be classified in terms of a space defined by the three dimensions of intelligence, agency and mobility (see Figure 1.1):

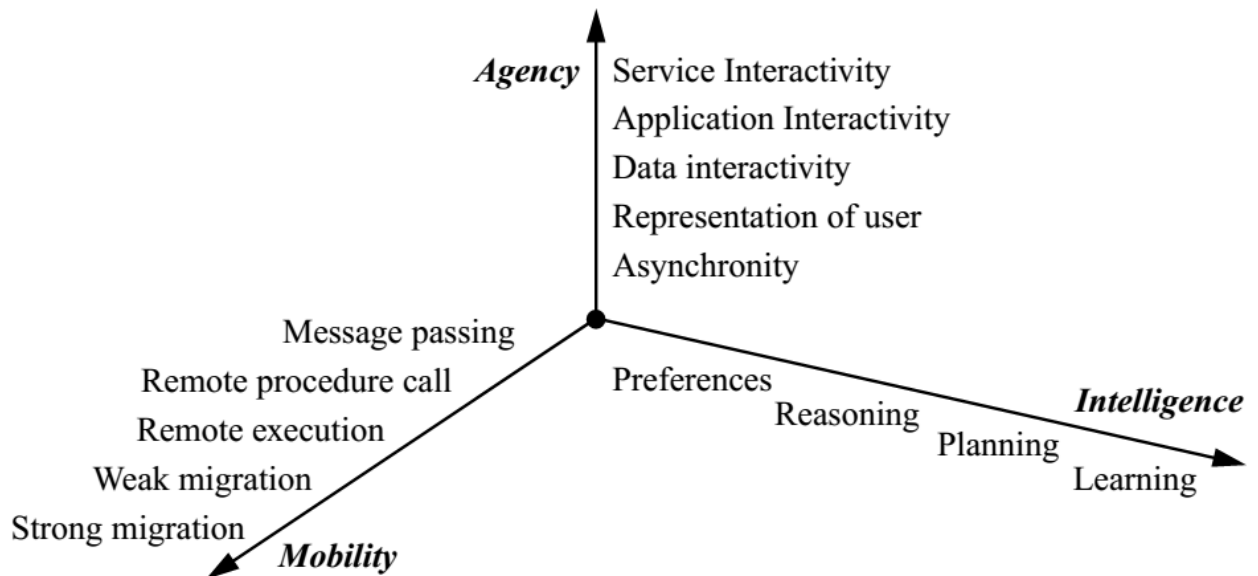


Figure 1.1 Space of Software Agents

2.3 Agent communication

2.3.1 The Knowledge Query and Manipulation Language (KQML)

KQML is an 'outer' language for agent communication. It defines an 'envelope' format for messages; an agent can explicitly state the intended illocutionary force of a message. KQML is not concerned with the content part of messages (Patil et al., 1992; Mayfield et al., 1996). [3]

2.3.2 The Knowledge Interchange Format (KIF)

KIF is a language explicitly intended to allow the representation of knowledge about some particular 'domain of discourse'. It was intended primarily (though not uniquely) to form the content parts of KQML messages. [3]

2.3.3 The FIPA agent communication languages

In 1995, the Foundation for Intelligent Physical Agents (FIPA) began its work on developing standards for agent systems. The center piece of this initiative was the development of an ACL (FIPA, 1999). This ACL is superficially similar to KQML: It defines

an 'outer' language for messages, it defines 20 performatives (such as inform) for defining the intended interpretation of messages, and it does not mandate any specific language for message content. In addition, the concrete syntax for FIPA ACL messages closely resembles that of KQML. [3]

3 Mobile agent

3.1 History

Mobile agents (MAs) were first proposed by White at General Magic in 1994. The concept of a mobile agent was implemented as the Telescript, while a runtime environment was also constructed for the execution and migration of MAs. The Telescript was patented in the U.S. in 1997. Chesset al. authors proposed “Itinerant Agents” in 1995, which are agents that are generated and dispatched by computers and that execute some management tasks over the network. [2]

3.2 Definition

Mobile agents have the ability to move from host to host, executing at each place and then keeping the results before moving to the next server. A simple picture of mobile agents in a network can be seen in figure 3. To be able to move, there must be an agent server, sometimes called agency, on each host that handles the incoming agents and executes them. This agency is also responsible for sending messages between agents and does some authentication if necessary. [18]

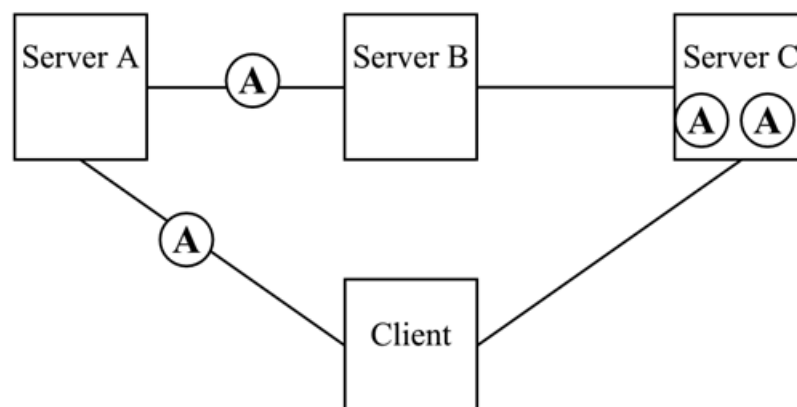


Figure 1.2 Mobile agents wander around in the network

3.3 Basic structure of a mobile agent

A mobile agent, as depicted in Figure 3, consists of three parts: code, state and data. The code is the aspect of the agent that is executed when it migrates to a platform. In the simplest case, there is a single code. The state is the data execution environment of the agent, including the program counter and the execution stack. This part is only found in agents that use strong migration. The data consists of the variables used by the agents, such as knowledge, file identifiers, etc. In weak migration this part is strictly necessary since agent code is constructed as a state machine with variables required to maintain state information. [3]

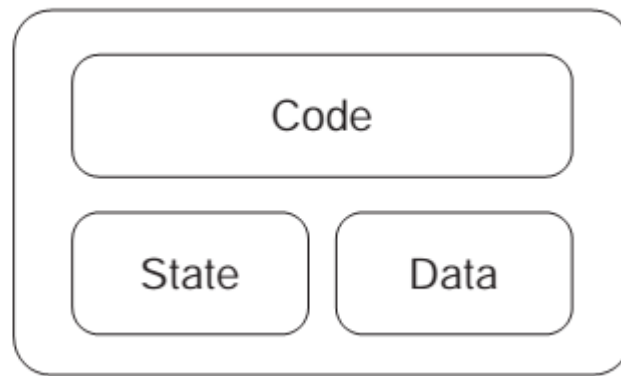


Figure 1.3 Basic structure of a mobile agent

3.4 Agent migration

The migration is a mechanism to continue the current execution of an agent at another location. In an agent system, the migration is initiated by the agent itself and not automatically by the system (e.g. for load balancing purposes). After the command to migrate is issued, the execution of the agent at the current location is stopped; the agent is removed from the source location and transported to the destination location. [19]

3.4.1 Weak and strength migration

The state of execution is migrated with the code so that computation can be resumed at the destination. According to the amount of detail captured in the state, we can classify agent migration into two types: strong and weak. [8]

a. Strong migration

It is the ability of an agent to migrate over a network, carrying the code and execution state, where the state includes the program counter, saved processor registers, and local

variables, which correspond to variables allocated in the stack frame of the agent's memory space, global variables. [8]

b. Weak migration

It is the ability of an agent to migrate over a network, carrying the code and partial execution state, where the state is variables in the heap frame, e.g., instance variables in object oriented programs, instead of its program counter and local variables declared in methods or functions. The agent is moved to and restarted on the destination with its global variables. The runtime system may explicitly invoke specified agent methods. Strong migration can cover weak migration, but it is a minority. This is because the execution state of an agent tends to be large and the marshaling and transmitting of the state over a network need heavy processing. Moreover, like the latter, the former cannot migrate agents that access the computational resources only available in current computers, e.g., input-and-output equipment and networks. [8]

3.5 Mobile agent properties

In order to get a better understanding of mobile agents, three properties will be defined and explained in this section. They are mobility, executability and autonomy. A mobile agent can be defined by using these three properties. This definition of three properties is also useful because each property can be analyzed regarding its security implications. A similar taxonomy has been made with slightly different properties. [2]

Mobile agents are actually a subset of the larger field of mobile code. A different way to classify mobile code is to divide its behavior into four distinct design paradigms. These are the well-known "Client-Server" paradigm, where the code does not move at all; "Remote Evaluation", which sends the code to another site where it is executed and the result is sent back; and "Code on Demand", where the code is downloaded from a distant site and executed on the local machine. The fourth paradigm is "Mobile Agents", which are small programs that wander around in the network. This classification system only helps us understand how mobile code works and where it came from. It is not very helpful when studying its security implications. [2]

3.5.1 Mobility

The word "mobile" comes from Latin and means something that moves quickly. At the other end of the scale, code moves very slowly when transported on a diskette or tape. It

moves somewhat faster when moved by the FTP mechanism. It is not slow because of the network, but because it often involves a manual operation like pointing and clicking or giving a command. If the transportation is without any manual work, it is only dependent on network speed. Note that a program code that needs to be compiled before it is executed is moving slower than an interpreted language, because compilation often requires manual intervention. This shows that there are degrees of mobility. The opposite of mobile is stationary. [2]

If something moves faster, it is usually more effective. Information is more effective when it moves and spreads faster. The same would apply to code that moves. Anything that moves fast also naturally becomes more dangerous. Stopping a virus attack that spreads from computer to computer within milliseconds is quite hard. For the data to be able to move, it needs a channel. A slow channel mentioned above is FTP. A fairly fast channel is WWW, but it requires that the user is active, except for some smart Java applets or JavaScript that keep the connection alive even when the user is inactive. Email is quite a fast channel because the emails are handled automatically by the mail servers, and one email can travel around the world in a couple of minutes. The fastest channel is of course direct TCP/IP, or UDP, which can be even faster in some cases. Sometimes in the literature and texts about mobile code you see the two notions, weak and strong mobility. [2]



Figure1.4 Code moves from site to site

3.5.2 Autonomy

When something is autonomous, it is acting on its own without any outside control. It is responding, reacting, or developing independently of the environment. A piece of program code and data is self-contained when it has encapsulated the code, data and environment into a unit. The environment is the current program state with its variables and bindings to resources. A binding can, for example, be a reference to a database or a local printer. [2]

An autonomous unit must also have a certain level of intelligence, so it can react properly to its environment. This often means some kind of rule-set. [2]

One example of computer programs acting on their own is automatic program updates where the program connects to a server, downloads new code and updates itself. This can, of course, be very practical, as you do not have to worry about the latest updates. [2]

There are differences in the intelligence in autonomous units, and the automatic updates are a simpler form of autonomy. Asynchronicity might also fit into this classification. Security implications. [2]

For each property described above, there are obvious security implications. The higher degree of mobility, executability and autonomy a piece of data has, the bigger threat it is to security. Fast-moving code can more effectively damage a networked computer system. Machine code can get closer to the operating system and kernel by calling system functions, or even replacing some parts of it. An autonomous piece of code can adapt to different environments and find better paths of attacks if the system is well guarded. [2]

If you can identify the threats, you can design proper protections against them. By slowing down the process of dealing with incoming data, you get better protection against an attack, for example requiring authentication before admitting the data. This is an unusual way of looking at the authentication process. [2]

It is harder to defend against the autonomy property as you need hosts that are smarter than the agents. One way is to fool the agents somehow, for example by giving wrong or incomplete data or special information that only your own agents can understand. [2]

3.6 Mobile agent advantages

- They reduce the network load by sending the code to the data host instead of sending the data over the network.
- They overcome network latency in real-time systems because they do not need a lot of bandwidth.
- They encapsulate protocols by bringing their own protocol code with them.
- They execute asynchronously and autonomously, and are therefore not dependent on a continuously open network connection.
- They adapt dynamically to new environments.
- They are naturally heterogeneous, and work well in a heterogeneous network.
- They are robust and fault tolerant. If a host shuts down, the agents can move to another host. They can also duplicate themselves and execute on several hosts in parallel.[18]

3.7 Disadvantages mobile agents

Nonetheless mobile agents are not the perfect paradigm and some issues cannot be overlooked. Next some technical and philosophical issues are summarized:

- ✚ The existing systems save network latency and bandwidth at the expense of higher loads on the service machines. The problem arises because most of the agents are written in relatively slow interpreted languages (most in Java), for portability and security reasons. The time saved avoiding intermediate network traffic is less than the time penalties from the slower execution and the migration overhead. Research groups have concentrated efforts in the compilation field achieving significant progress with new techniques such as just-on-time compilation and software fault isolation, that execute nearly as fast as native compiled code. Other research groups are working in reducing the migration overhead. With all these techniques available the accepting and execution of a mobile agent will be must faster than it is now, and nearly as fast as a native compiled procedure. [7]
- ✚ Almost all the mobile agent's systems allow an agent to freely move among heterogeneous machines. The code is compiled to a platform independent code (usually Java byte code) that upon its arrival is interpreted or compiled to native code. However, for mobile agents to be widely used the code must be portable across mobile agent's systems. This will require a huge standardization effort, being the OMG MASIF standard the first step. The trend points to a standard execution environment and a standard encoding format for the agent's code and state. [7]
- ✚ Today it is possible to implement a mobile agent that protects a machine from malicious agents. Nonetheless some other security issues remain as protecting the machines without artificially limiting agent access rights; protecting an agent from malicious machines; and protecting groups of machines that are not under single administrative control. The solution to these problems must not limit the use of mobile agents in a truly open environment such as the Internet. Some research groups are working in this field and it is probable that in a few years mobile agents will be completely secure. [7]
- ✚ The mobile agent's paradigm does not have a killer application. It can be used to implement applications which lead to faster performances in several cases, but these applications can be implemented just as cleanly and efficiently with traditional techniques. Although different techniques must be applied for different applications. So, the advantages of the mobile agent paradigm are not many when an application is considered alone. The mobile agent community must present a set of applications and argue that the entire set can be implemented with less effort and less programming groups, using mobile agents. It is unreasonable that any Internet service will be willing

to change radically from the client/server paradigm to the mobile agent paradigm. An evolutionary path from current systems to mobile agent based systems must be provided. [7]

- ✚ The problem of advertising must not be ignored as well. Many web sites earn money only from the advertisements. If these sites allow free access to their information to mobile agents, the number of persons visiting the site will presumably decrease. [7]

4 Mobile agent applications

4.1 Electronic commerce

Mobile agents are well suited for electronic commerce. A commercial transaction may require real-time access to remote resources such as stock quotes and perhaps even agent-to-agent negotiation. Different agents will have different goals, and will implement and exercise different strategies to accomplish these goals. We envision agents that embody the intentions of their creators, and act and negotiate on their behalf. Mobile agent technology is a very appealing solution to this kind of problem. [20]

4.2 Personal assistance

The mobile agent's ability to execute on remote hosts makes it suitable as an "assistant" capable of performing tasks in the network on behalf of its creator. The remote assistant will operate independently of its limited network connectivity, and the creator can feel free to turn off his or her computer. To schedule a meeting with several other people, a user could send a mobile agent to interact with the representative agents of each of the people invited to the meeting. The agents could negotiate and establish a meeting time. [20]

4.3 Distributed information retrieval

Information retrieval is an often-used example of a mobile agent application. Instead of moving large amounts of data to the search engine so that it can create search indexes, you dispatch agents to remote information sources, where they locally create search indexes that can later be shipped back to the origin. Mobile agents are also able to perform extended searches that are not constrained by the hours during which the creator's computer is operational. [20]

4.4 Telecommunication networks services

Support and management of advanced telecommunication services are characterized by dynamic network reconfiguration and user customization. The physical size of these networks and the strict requirements under which they operate call for mobile agent technology to form the "glue" that keeps such systems flexible yet effective. [20]

4.5 Workflow applications and groupware

It is in the nature of workflow to support the flow of information between co-workers. The mobile agent is particular useful here since, in addition to mobility, it provides a degree of autonomy to the work flow item. Individual workflow items fully embody the information and behavior needed for them to move through the organization independent of any particular application. [20]

4.6 Monitoring and notification

This is one of the "classical" mobile agent applications that highlight the asynchronous nature of mobile agents. An agent is able to monitor a given information source without being dependent on the location from which it originates. Agents can be dispatched to wait for certain kinds of information to become available. It is often important that monitoring agents have life spans that exceed or are independent of the computing processes that create them. [20]

4.7 Information dissemination

Mobile agents embody the so-called Internet "push" model. Agents are able to disseminate information such as news and automatic software updates for vendors. The agents will bring the new software components as well as the installation procedures directly to the customer's personal computer and will autonomously update and manage the software on the computer. [20]

5 Agent architecture vs. client/server architecture

In the traditional client server architecture, all connections to server are initiated from the client. The advantage is that these connections can be managed in parallel (Figure 5). With an agent architecture (Figure 6 the network is dynamic, and the client does not have to know the structure of the network. The fact that agents communicate with a high-level communication language and work over a logical network makes them more reliable as the logical network

which is dynamic will adapt itself to the current conditions and be less affected by network failure.

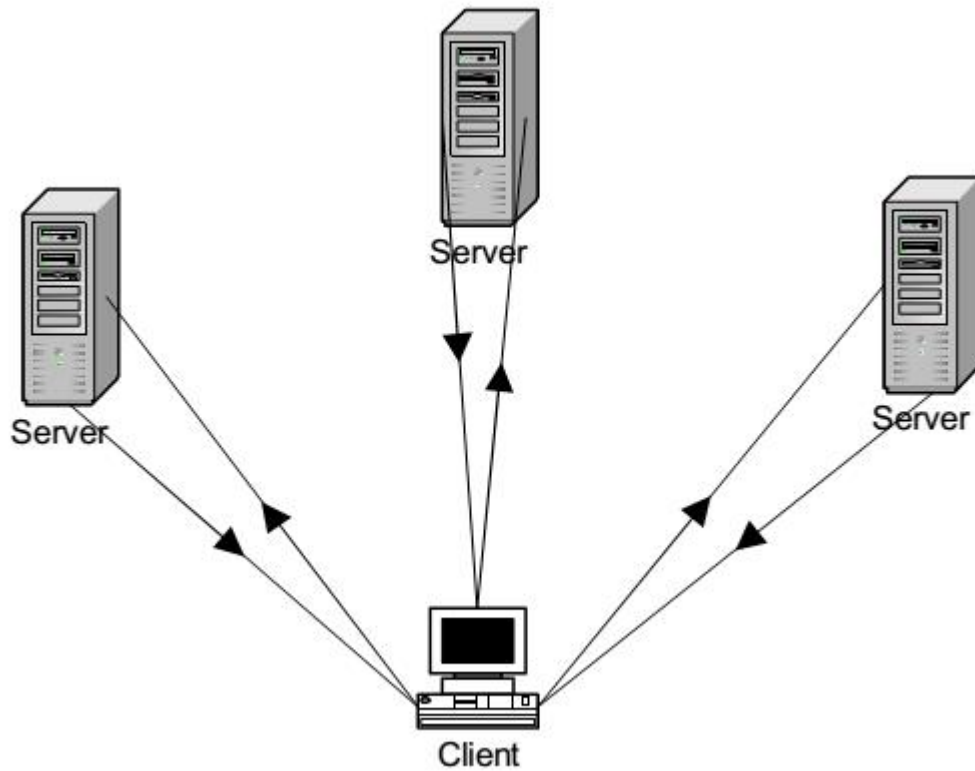


Figure 1.5 Traditional client/server communication

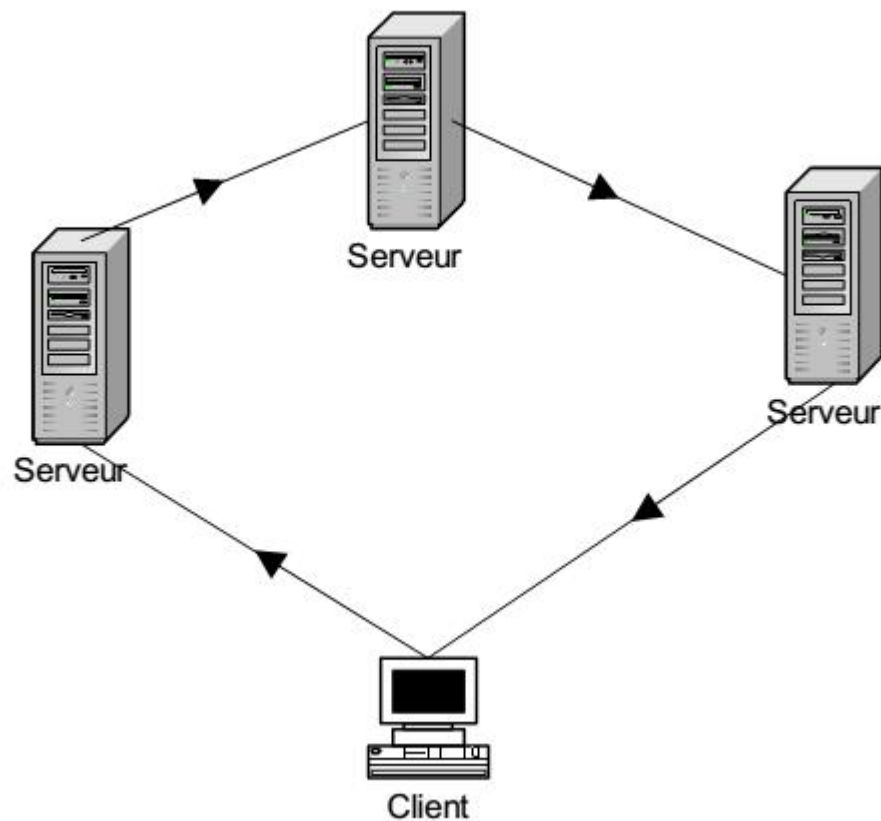


Figure 1.6 Communication with the use of mobile agents

Mobile agents differ from traditional client/server application by moving themselves where the data are, instead of moving the data to where the application resides. This improves performance of the data collection by requiring less bandwidth. Another advantage is the fact that mobile agents can perform their task asynchronously, or offline: the host of origin can initiate agents, tell them to migrate, go offline, and when back online, waiting for agents to come back. [24]

6 Mobile Agent Standardization: MASIF

Some aspects of mobile agent technology must be standardized. The companies Crystaliz, General Magic Inc., GMD Fokus, IBM Corporation, and the Open Group have jointly developed a proposal for a Mobile Agent System Interoperability Facility (MASIF) and brought it to the attention of the Object Management Group (OMG).

MASIF addresses the interfaces between agent systems, not between agent applications and agent systems. Even though the former seems to be more relevant for application developers, it is the latter that allows mobile agents to travel across multiple hosts in an open environment. MASIF is clearly not about language interoperability. Language interoperability

for mobile objects is very difficult and MASIF is limited to interoperability between agent systems written in the same language, but potentially by different vendors. Furthermore, MASIF does not attempt to standardize local agent operations such as agent interpretation, serialization, or execution. You can say that MASIF defines the interfaces at the agent system level rather than at the agent level. [20]

7 Contemporary Mobile Agent Systems

So what kind of mobile agent systems are available for you? Fortunately, Java has generated a flood of experimental mobile agent systems. Numerous systems are currently under development, and most of them are available for evaluation on the Web. [20]

We will, mention a few interesting Java-based mobile agent systems: Aglets, Odyssey, Concordia, and Voyager. [20]

7.1 Aglets

It is the mirror the applet model in Java. The goal was to bring the flavor of mobility to the applet. The term aglet is indeed a portmanteau word combining agent and applet. [20]

7.2 Odyssey

General Magic Inc. invented the mobile agent and created the first commercial mobile agent system called Telescript. Being based on a proprietary language and network architecture, Telescript had a short life. In response to the popularity of the Internet and later the steamrolling success of the Java language,

General Magic decided to re-implement the mobile agent paradigm in its Java-based Odyssey. This system effectively implements the Telescript concepts in the shape of Java classes. The result is a Java class library that enables developers to create their own mobile agent applications. [20]

7.3 Concordia

Mitsubishi's Concordia is a framework for the development and management of mobile agent applications which extend to any system supporting Java. Concordia consists of multiple components, all written in Java, which are combined together to provide a complete environment for distributed applications. A Concordia system, at its simplest, is made up of a standard Java VM, a Server, and a set of agents. [20]

7.4 Voyager

Object Space's Voyager is a platform for agent-enhanced distributed computing in Java. While Voyager provides an extensive set of object messaging capabilities, it also allows objects to move as agents in the network. You can say that Voyager combines the properties of a Java-based object request broker with those of a mobile agent system. In this way, Voyager allows Java programmers to create network applications using both traditional and agent-enhanced distributed programming techniques. [20]

We would like to note that the Java-based mobile agent systems have a lot in common. Beside the programming language, they all rely on standard versions of the Java virtual machine and Java's object serialization mechanism. A common server based architecture permeates all the systems. Agent transport mechanisms and the support for interaction (messaging) varies a lot. [20]

Although a majority of the contemporary mobile agent systems are based on the Java language system, you will also be able to find other languages in use. Most significant languages are Tcl and Python. [20]

7.5 Agent Tcl

This is a mobile agent system whose agents can be written in Tcl. Dartmouth College's Agent Tcl has extensive navigation and communication services, security mechanisms, and debugging and tracking tools. The main component of Agent Tcl is a server that runs on each machine and that allows the entire execution state including local variables and instruction pointer to move. When an agent wants to migrate to a new machine, it calls a single function, agent-jump, which automatically captures the complete state of the agent and sends this state information to the server on the destination machine. The destination server starts up a Tcl execution, loads the state information into this execution environment, and restarts the agent from the exact point at which it left off. [20]

7.6 Ara

Tcl-based Ara from University of Kaiserslautern is a platform for the portable and secure execution of mobile agents in heterogeneous networks. The research project is primarily concerned with system support for general mobile agents regarding secure and portable execution, and much less with application-level features of agents, such as agent cooperation patterns, intelligent behavior, and user modeling. [20]

7.7 TACOMA

The TACOMA project focuses on operating system support for agents and how agents can be used to solve problems traditionally addressed by operating systems. The TACOMA system is based on UNIX and TCP. The system supports agents written in C, Tcl/Tk, Perl, Python, and Scheme (Elk). The system itself is implemented in C. [20]

8 Conclusion

There are many types of agents according to their properties, but Mobile agent is the most popular one because its ability of mobility which is very important in internet and security like intrusion detection. In this last years, the old Intrusion Detection Systems (IDS) face some problems because of its limitation. In the next chapter, we will give information about IDS and its characteristics, architecture, advantages and its limitations.