DESIGN AND IMPLEMENTATION OF SOLAR NETWORK MANAGEMENT SYSTEM FOR COMPUTER BASE TEST

BY

MOHAMMED ALIYU
ND/23/COM/PT/0352

A PROJECT SUMMITTED TO

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY (IICT) KWARA STATE POLYTECHNIC, ILORIN

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF NATIONAL DIPLOMA (ND) 2023/2024 SESSION

2024/2025 SESSION

CERTIFICATION

This is to certify that this project research was carried out by **MOHAMMED ALIYU** with Matriculation Number **ND/23/COM/PT/0352,** has been read and approved as meeting part of the requirements for the award of National Diploma (ND) in Computer Science.

_____	_____
MR. QUADRI B. A.	**DATE**
(Project supervisor)	
_____	_____
MR. OYEDEPO, F.S.	**DATE**
(Head of Department)	
_____	_____
External Supervisor	**DATE**

DEDICATION

This project is dedicated to Almighty the most Beneficient, the Most merciful and to the Department of Computer Science Kwara State Polytechnic, Ilorin

ACKNOWLEDGEMENT

All adoration and glory are due to the Almighty for His Love, guidance and protection. I thank him for the perfect health, wisdom, knowledge, strength and opportunity to complete this stage of Academic (National Diploma) in peace.

My Profound gratitude goes to my amiable supervisor Mr. Quadri R.A, thanks so much for your support and care throughout the supervision of this research work, also to my good H.O.D Mr. Oyedepo and to all the entire staff of the Institute of Information and Communication Technology.

Finally, I express my gratitude to my parents Mr. and Mrs. Mohammed, friends and loved ones who supported us in one way or the other.

TABLE OF CONTENTS

ABSTRACT

As the need for information is becoming more necessary every day, it has become necessary to decentralize and make independent the method of data access and data processing by the use of computer networking. As networks grew, large networks cannot be monitored and managed by human efforts alone. Also with the rapid growth of the Internet, network management is a more demanding domain. In fact, Network Administrators become overwhelmed with a lot of simple and boring tasks. But contemporary network management systems as represented by Simple Network Management Protocol (SNMP) are based on client-server centralized model which may lead to inefficiency when the size of networks grow larger. This could be overcome by the use of intelligent multi-agent software. This research work first investigated the possibilities and strategies of network management and monitoring with intelligent multi-agent technologies. A research survey and detailed evaluation of existing platforms available today were presented and then finds a cooperative platform. JADE (Java Agent DEvelopment framework) was selected as the software framework that facilitates development of interoperable intelligent multi-agent systems and that is distributed under an Open Source License. The thesis also presents JADE, details of its architecture, its technological components together with its implementation that allows good runtime efficiency, software reuse, agent mobility and the realization of different agent architecture. Finally, a distributed, multi-threaded prototype that implements simulated network monitoring and management system was proposed using Visual Basic programming language.

CHAPTER ONE
GENERAL INTRODUCTION

1.1 BACKGROUND TO THE STUDY

Network Management comprises all the measures necessary to ensure effective and efficient operations of a networked system. It is a critical discipline whose main objective is to keep the communication services of organizations working properly in order to exchange information in a fast and reliable way [1]. Intelligent is a general term to describe some of mind activities such as judging, logical thinking, planning and learning [2, 10]. Artificial Intelligence (AI) is the study of how machines could be programmed to emulate intelligent behaviours especially that of human [6]. It also attempts to bring computers a little closer to human brain's capabilities by imitating certain aspects of information processing in the brain, in a highly simplified way. An agent is anything that can perceive its environment through sensors and acting upon that environment through effectors [2, 6]. The term agent is becoming very popular in the computing press as it is within the artificial intelligence and the computer science communities [9].

Agent software has continued to attract significant research attention due to their intelligence, mobility and collaborative nature. Agent-based systems play variety of roles in the development of various essential industrial applications such as manufacturing, entertainment, electronic commerce, user assistance, user interface design, service and business management, information retrieval and network management [3]. The agent entities can roam the Internet and seek out information and services of interest to users [6]. They are also expected to mimic human behaviours and perform flexible autonomous action on behalf of the users in order to meet their design objectives. Agents are also expected to know users' priorities, taste and should be able to take initiative on users' behaviour [4, 8].

There is high need of agents in computing because of rapid development in computer and data-networking technology to support those needs, and explosion in the variety of equipment and networks. A multi-agent system (MAS) is a system composed of multiple interacting agents. Multi agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve [7].

However, with rapid growth of the networks and the Internet, large networks cannot be monitored and managed by human efforts alone [9]. Contemporary network management systems as represented by Simple Network Management Protocol (SNMP) are based on traditional client-server centralized paradigm which may be inefficient when the size of networks grow. Therefore, the complexity of such system dictates the use of intelligent multi-agents for network management and monitoring. Intelligent multi-agent technologies are leading solution, among other current technologies such as Remote Monitoring (RMON) and Management by Delegation (MbD) for achieving complex network management goals [6, 8]. This is because agent-based technology has the capability to distribute intelligence throughout the network and dynamically perform network management functions [9].

To perform network management with intelligent agents, a multi-agent platform is needed [4]. Platforms are software components that can range from a collection of classes compatible among them to high-level programming environment equipped with several tools helping programming agents [4]. A lot of research and application on agent technologies were investigated and presented. There are many interesting platforms available today that support intelligent agents, such as JATlite, JADE, JAFMAS, Aglets, and Odyssey etc. A detailed evaluation of some of these platforms available today were presented and JADE (Java Agent DEvelopment Framework) was selected for the implementation of network monitoring and management. JADE is a software framework to

facilitate the development of inter-operable intelligent multi-agent systems that is used by a heterogeneous community of users as a tool for both supporting research activities and building real applications [30].

1.2 PROBLEM DEFINITIONS

The network management system as represented by Simple Network Management Protocol (SNMP) is based on client-server centralized model. It was widely adopted by the Internet Protocol (IP) world to manage local area networks, wide area networks and Intranets, and to a lesser extent, to manage distributed systems [9]. However, the following are some of the shortcomings of SNMP for network management;

1. Though Simple Network Management Protocol (SNMP) is the most pervasive protocol, it lacks the scalability, reliability, flexibility and adaptability necessary to effectively support large and complex networks [5, 8].

2. In SNMP, the central station collects and analyses data retrieved from physically distributed network elements [9, 8]. But network system based on client-server model normally requires transferring large amount of management data between the network manager and agents. These require high bandwidth and also cause a processing bottle-neck for the network manager. The problem becomes more complex as networks grow larger. An attempt to scale their technologies to users' magnitude can be extremely difficult and very costly [5, 8, 9].

1.3 AIMS AND OBJECTIVES

The major aim of this research work is to start study from the Distributed Artificial Intelligence (DAI) perspective, investigate on what has been done there, and then attempt to apply this to network management. The objectives are as follows:

a. To investigate the possibilities and strategies of network management and monitoring with intelligent multi agents technologies, and to facilitate the effort of network management according to established policies.

b. To evaluate the existing platforms available today that support distributed, collaborative and intelligent agent and then propose the most suitable platform among them for the implementation of intelligent multi-agents system.

c. To design a distributed, multi-threaded prototype that implement simulated network monitoring and management using Visual Basic Programming language.

1.4 SIGNIFICANCE OF THE STUDY

Multi-agent technology is applied by intelligent systems to solve the problems of analysis of complex systems and intelligent management activities [10]. Agents are sophisticated computer programs that act autonomously on behalf of their users, across open and distributed environment to solve growing number of complex problems. A multi agent system has been described as loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver [2]. The following are major benefits of the study;

a. A multi-agent network management system assists the network administrators to do a lot of simple and boring tasks.

b. It makes life easier, save time and simplify the growing complexity of world.

c. A multi agent network management system acts as an assistant to network administrator by learning from what the administrator likes and can anticipate what needs to be done at a particular time.

d. It reduces the amount of bandwidth requires for the transfer of large amount of data which could cause processing bottleneck.

1.5 SCOPE OF THE STUDY

The scope of this dissertation covers the investigation of the possibilities of using intelligent multi-agent in network monitoring and management. I later studied the existing platforms available today and find the most suitable one among them. JADE was then considered among others, as a middle-ware that implements an efficient agent platform and supports the development of multi-agent systems. Detailed architecture of JADE, its technological components together with its implementation that allows good runtime efficiency, software reuse, agent mobility and the realization of different agent architecture were discussed. An intelligent multi-agent based network management system was implemented using JADE agent platform. A multi-threaded prototype was designed using Visual Basic programming language for network monitoring and management. Other areas of network management such as fault, security and configuration management were beyond the scope of this research work.

1.6 RESEARCH METHODOLOGY

The search for literatures for this dissertation was performed mainly with reading existing project work, journal papers, proceeding of conferences, text books on agent technologies, and browsing the World Wide Web (www). Investigation on the existing platform for agent development and multi agent designed tools was done mainly on the websites of each platform on World Wide Web. However, the study draws on the theory, experimentation and findings of the various architectures that have been published. The overview of the proposed agent platform architecture was presented.

CHAPTER TWO
LITERATURE REVIEW

2.1 REVIEW OF PAST WORK

Intelligent agents have originated a lot of discussion about what they are, and how they are use, and how they are different from general program. They have great potentials to use in different purposes and research areas. Michaud (1998), in his project titled “Intelligent Agents for Network Management” compiled a state of the art on cooperative platforms based on intelligent agents and selected one of those platforms and experimented with the management of IP multimedia network. He stated some areas of application of multi agents such as manufacturing, electronic commerce, user assistance, service and business management, and information retrieval. He aimed at experiment with the strongly distributed cooperative paradigms with multi agent system and his scope was to concentrate on managing multimedia services and networks. He chose Java Agent Framework Multi Agent System (JAFMAS) out of many platforms studied because it relieves the developer from the effort of programming cooperation mechanism from scratch.

Richards (2003) examined “Intelligent-Agent-Based Management of Heterogeneous for Army Enterprises”. He found that the army was undergoing a major realignment in accordance with Joint Vision 2010/2020 transformation to establish an enterprise command that is the single authority to operate and manage the army enterprise information infrastructure. He noted that there are a number of critical network management issues that the army will have to overcome before attaining the full capabilities to manage the Ml spectrum of army networks at the enterprise level. Moreover, he stated that most of the network management architectures in the army are based on traditional centralized network management approach such as Simple Network Management Protocol (SNMP) which is

not effective to support an enterprise like large and complex network of the army. He made the argument that intelligent-agent based technologies are the leading solution among other current technologies to achieve army's enterprise network management goals. He studied and discussed the characteristics of ten platforms. He later chose CoABS platform and described it in greater detail. The program was aimed at developing and demonstrating techniques to safely control, coordinate and manage large systems of autonomous software agents.

Luo (2002) conducted a research work on Agent Based Network Management System. He noticed that contemporary network management systems as represented by Simple Network Management Protocol (SNMP) are based on the client-server centralized paradigm which may lead to inefficiency when the managed networks are large in scale. He aimed at solving the above mentioned problems by implementing a mobile agent based network management system using Java Agent DEvelopment framework (JADE) platform. Applications implemented in his system were network element's status monitoring, SNMP table filtering and global filtering. He also considered and investigated the security problem of using mobile agent.

Andrey (1998) in his project titled "Agent Based Network Management" investigated some possibilities and some strategies of network management by intelligent agents. He focused on connection admission control for multimedia sessions (like video conferences), by collaborative agents coordinated by market based mechanisms. He later focused on two platforms LALO and Plangnet for network management.

Based on the above literatures and my findings, it was discovered that JAFMAS was not presently available having search through it web site address several times. However, many researchers have not been used JADE for the network management and monitoring,

and then I decided to do more research on JADE and its implementation. This is because the platform is readily available on its web site for onward downloading for research and academic work.

2.2 REVIEW OF GENERAL TEXTS

Many published studies focus on network management with intelligent multi agent technologies. Network management usually refers to the management of a network's physical infrastructure: hubs, switches, routers, and gateways [24]. Network management comprises all the measures necessary to ensure effective and efficient operations of a network system [5]. There are several approaches that have been studied to tackle these requirements, but intelligent agent is the most interesting and promising [3]. Intelligent agent is an agent that can adapt to the needs of different users, it can learn new concepts and techniques, it can anticipate the need of the user and it can take initiative and make suggestions to the user. The major goal of an intelligent agent is to reduce user's work and information overloading. Intelligent agents can provide services in filtering data, searching for information, online tutoring and so on [23].

Madejski (2007) noted that agent-based approach provides, a convenient way to model and implement manufacturing enterprise integration and supply chain management set up as a network of cooperating, intelligent agent. Ali et al (2010) discovered that multi agent systems are used in such applications and problem solving that are distributed in nature. It decomposes task into several subtasks, and the subtasks are distributed among various agents, while the agents interact with each other. According to Villa et al (2007), telecommunication networks have become an inseparable technology; however there is a noticeable lack of powerful and dynamic management tools for resource configuration mainly due to rapid increase in transmission speeds and increasing number of users and services. They therefore noted that there is need for dynamic resource management system

that can maximize network utilization and fast restoration mechanisms. A multi-agent system is a good way of improving network management because it is an inherently distributed solution and it introduces artificial intelligence based techniques.

Intelligent Agents are autonomous because they function without requiring that the console or management server be running. An Agent that services a database can run when the database is down, allowing the agent to start up or shut down the database. The Intelligent Agents can independently perform administrative job tasks at any time, without active participation by the administrator. Similarly, the Agents can autonomously detect and react to events, allowing them to monitor the system and execute a fix job to correct problems without the intervention of the administrator (Oracle, 2002).

2.3 NETWORK MANAGEMENT SYSTEM

Network Management System (NMS) can formally be defined as the controlling of a complex data network so as to maximize its efficiency and productivity that involves active and passive monitoring of network resources for the purpose of troubleshooting, detecting potential problems, improving performance, reporting and documentation [12]. It comprises all the measures necessary to ensure effective and efficient operation of a network system [5]. Network management is a critical discipline whose main objective is to keep the communication services of an organization working properly in order to allow computers to exchange information in a fast and reliable way. Neglecting network management usually results in weak networks and, most important economic losses due to communication failures [1].

Network Management System (NMS) is becoming increasingly complex due to the embedded increasing complexity of the network elements in addition to increase in the size of the network, the needs of more sophisticated services, and the heterogeneity challenges

[9]. The management of large data networks, such as a national WAN, is without doubt a complex and cumbersome task. Existing network management software cannot meet the requirements of the constantly increasing size and complexity of today's TCP/IP based networks, since in most cases it provides only simple monitoring tools. Therefore, the full exploitation of a WAN cannot easily be achieved without user- friendly, intelligent network management software, enhanced with diagnostic and decision support services [9, 19]. The International Organization for Standards has defined Network Management as consisting of five key areas. These areas and their underlying functions are listed below [5, 15, 16]:

(i) Fault Management: Fault management is the process of detecting and correcting network problems. It involves recovery, and documentation of network anomalies and failures. Faults typically manifest themselves as transmission errors or failures in the equipment or interface.

(ii) Configuration Management: The configuration management functions detect and control the state of the network resources. This entails the initialization, modification, and shutdown of a network. It records and maintain network configuration, update configuration parameters to ensure normal network operations.

(iii) Accounting Management: Accounting management functions collect and process resource consumption data. It involves monitoring the login and logoff records, and checking the network usage. It includes user management and administration, billing on usage of network resources and services.

(iv) Performance Management: Performance management involves measuring the performance of a network and its resources in terms of utilization, throughput, error rates, and response times. With performance management information, a network-manager can reduce or prevent network overcrowding and inaccessibility. This provides reliable and high quality network performance. This includes quality of

service provision and regulating crucial performance parameters such as network throughput, resource utilization, delay, congestion level, and packet loss.

(v) Security Management: Security management deals with ensuring overall security of the network, including protecting sensitive information through the control of access points to that information; for example, blocking unauthorized access to database records. It provides protection against all security threats to network resources, its services, and data. In addition, ensure user privacy and control user access rights.

2.4 THE NEEDS FOR NETWORK MANAGEMENT

A network management system (NMS) consists of managers, agents, and dual-role entities with both manager and agent capabilities. In conventional systems, a manager is in charge of performing management functions whereas an agent normally performs simple tasks such as acquiring data. Nowadays, however, agents are performing tasks with more responsibilities, blurring the clear distinction between managers and agents [22]. There are number of reasons for network management in an organization, some of which are [9]:

i. Balancing various needs: The information and computing resources of an organization must provide a spectrum of end users with various applications at given levels of support, with specific requirements in the areas of performance, availability, and security. The network manager must assign and control resources to balance these various needs.

ii. Controlling complexity: The continued growth in the number of network components, end users, interfaces, protocols, and vendors threatens management with loss of control over what is connected to the network and how network resources are used.

iii. Controlling costs: Resources utilization must be monitored and controlled to enable essential end-user needs to be satisfied with reasonable cost.

iv. ***Controlling corporate strategic assets:*** Networks and distributed computing resources are increasingly vital resources for most organizations. Without effective control, these resources do not provide the payback that corporate management requires.

v. ***Improving service:*** End users expect the better and improved service as the information and computing resources of the organization grow and distribute.

vi. ***Reducing downtime:*** As the network resources of an organization become more important, minimum availability requirements approach 100 percent. In addition to proper redundant design network management has an indispensable role to play in ensuring high availability of its resources.

2.5 NETWORK MANAGEMENT PROBLEMS

As networks grow, large networks cannot be monitored and managed by human efforts alone and with the growing size of the networks and rising number of network platforms and devices, many disparate network management solutions worked there way into those organizations. Many organizations experienced a myriad of network management problems due to issues such as multi- vendor interoperability, lack of expertise, and reliability [5, 9]. Network Management (NM) is becoming increasingly complex due to the embedded increasing complexity of the network elements in addition to the increase in size of the network, the needs of more sophisticated services, and the heterogeneity challenges. However, new network management system must be [3]:

- Flexible to deal with the fast and frequent configuration changes
- Scalable to improve the profitability of the huge investment that represent its installation, configuration and launching.
- Reliable and therefore able to anticipate the risk of failure and to continue to work even if part of the network is cut out of the network management system.

- Simple to decrease the learning delay, reduce the human expertise needed to use it and secure.

2.5.1 PROBLEMS OF CENTRALIZED MANAGEMENT APPROACH

Communication networks are traditionally managed by centralized management systems. A centralized management system typically contains one manager. This manager may control a potentially large number of network elements by manipulating local management agents. This can be viewed as client-server approaches where the network management system is a client to these management agents. The usage of centralized network management approach has a considerable number of disadvantages. The most important ones are described below [22]:

a) The use of central entity in charge of polling all network elements has a negative effect on the scalability. The main problem in the centralized approach is scalability. A large amount of nodes will lead to a large polling load of the network. Especially links near the NMS will have to handle all these requests and responses. Furthermore, since all management logic is concentrated in the NMS and the agents simply collect the data, only the NMS can analyze all the data. Both a traffic bottleneck and a processing bottleneck at the NMS will then be unavoidable.

b) Using a single server that is in charge of network management and contains all management logic introduces a *single point of failure.* Naturally, redundancy can be created by server replication for example, but this is usually a costly operation. In situations where the central entity becomes unavailable, due to link failures or a (temporal) outage for example, management operations will not be executed.

The increase of the size of networks in the 1990s and the wide availability of the IP stack on network devices influenced the need for other approaches [22]. As a result of increase in the size of the networks and the networks become more complex to manage, the Common Management Information Protocol (CMIP) published by the International Organization for Standardization (ISO) and the Simple Network Management Protocol (SNMP) published by the Internet Engineering Task Force (IETF) was introduce in the late 1980s to overcome these problems.

2.6 SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

Network management protocols were developed so that the process of network management could be automated as much as possible. The Simple Network Management Protocol (SNMP) is an application layer protocol used to manage network resources. This standardization gives network administrators the ability to monitor network performance [26]. With the increase in the size and number of computer networks, the need for efficient management of resources has emerged as a pressing issue for network administrators. Administrators are constantly maintaining their networks in order to maximize efficiency. The Simple Network Management Protocol was developed to assist in network resource management [26]. SNMP is the most popular protocol used to manage networked devices. It was designed in the late 1980s to facilitate the exchange of management information between networked devices operating at the application layer of the ISO/OSI model [25].

SNMP is formally defined in RFC 1157:

Implicit in the SNMP architectural model is a collection of network management stations and network elements. Network management stations execute management applications which monitor and control network elements. Network elements are devices such as hosts, gateways, terminal servers, and the like, which have management agents responsible for performing the network management functions requested by the network management

stations. The Simple Network Management Protocol (SNMP) is used to communicate management information between the network management Stations and the agents in the network elements [26].

SNMP is used to manage a variety of network resources including hardware products such as servers, printers, PCs, or networking products, or software such as the Windows NT operating system or a database application. Using SNMP management systems, network administrators can browse the configuration of a device, monitor collected variables such as network packet counts, or receive SNMP traps, a message sent from the agent to the manager when an "event, such as a power failure occurs in a system [15, 26].

There are three versions of SNMP these are: SNMP version 1 (SNMPv1), SNMP version 2 (SNMPv2) and SNMP version 3 (SNMPv3). SNMPv1 was easy to implement but had numerous security problems. SNMPv2 was developed to enhance security and functionality, but was still lacking features in security authentication and encryption. SNMPv3, which was designed to be backward compatible with the first two versions, addresses these concerns by including access control, authentication, and privacy of management information.

2.6.1 FUNCTIONALITY

A system of network components works together to form the functionality of SNMP, the SNMP has three basic components:

i. Structure of Management Information (SMI)
ii. Management Information Base (MIB)
iii. SNMP agents

i. **Structure of Management Information (SMI):** The SMI defines the data types that are allowed in the MIB. It sets aside a unique naming structure for each managed object. Typically MIB objects have six attributes. An object will have a name, an object identifier, a syntax field, an access field, a status field, and a text description.

ii. **Management Information Base (MIB):** The MIB is a collection of network information. This information is stored in a database of managed objects that can be accessed using network managing protocols such as SNMP. A managed object can represent a characteristic of a certain managed device. The MIB object may hold a value associated with the number of packets that have come in since the last system reset. It may store the number of clock ticks since the last reset or even a specific administrative state of a device. These values are stored in scalar and tabular forms. Scalar objects define a singular object instance. A tabular object defines a group of object instances that are found in MIB tables.

iii. **SNMP Agents:** All network devices that are to be SNMP managed need to be fitted with an agent that executes all the MIB objects that are relevant. The agent provides the information contained in the MIB to management applications when asked.

2.6.2 SNMP ARCHITECTURE

An SNMP-managed network includes management stations and network devices. The management stations execute management applications like SNMP, which monitor network performance. Network agents are responsible for maintaining network statistics for management stations. When asked, each managed network device is expected to communicate such information for processing. SMI enables a vendor to write an SMI-compatible management object. This object is run through a MIB compiler to create

an executable code. The code is installed in network devices and management consoles that in turn generate network reports.

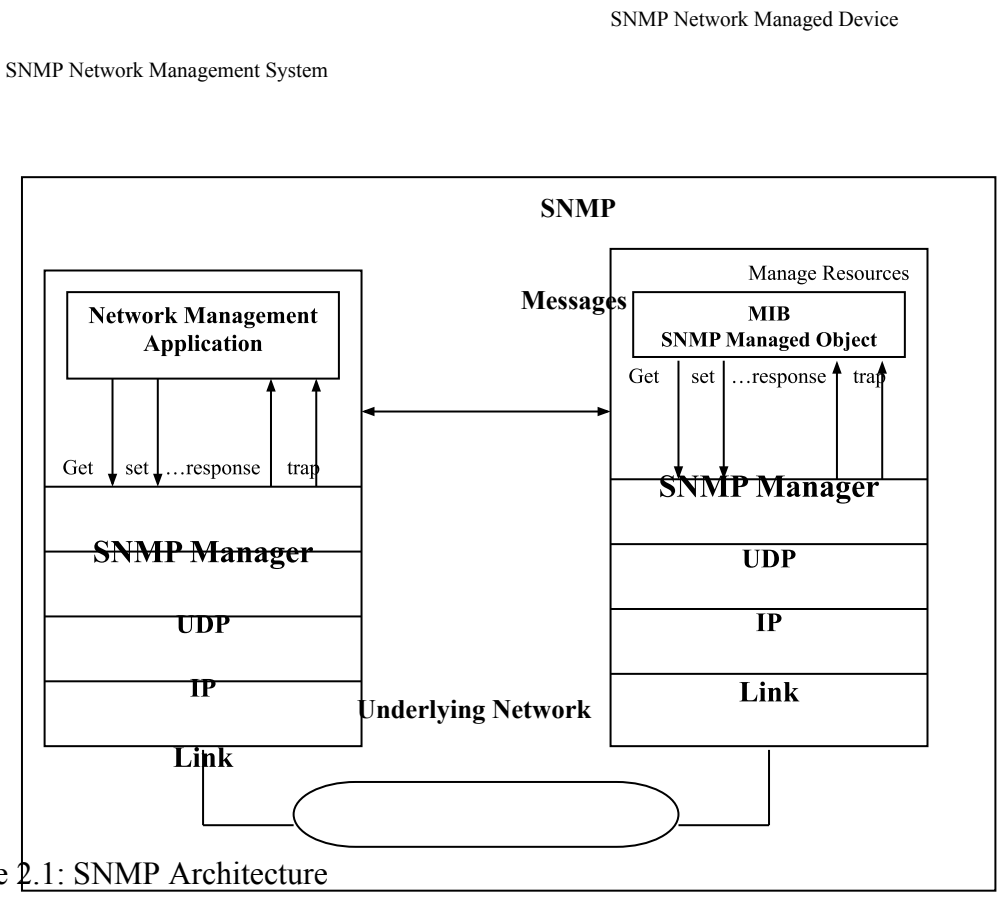

Figure 2.1: SNMP Architecture

2.6.3 ADVANTAGES OF SNMP

Implementation of a SNMP-compliant network offers significant benefits. These benefits allow a network administrator control in managing a healthy and efficient network.

i. **Control:** The benefits of running an SNMP-compliant application include the abilities to prevent, detect, and correct network-related issues. SNMP is easy-to-use

and allows administrators the control they need to maintain a healthy network. It provides administrators with a network management mechanism that efficiently monitors network performance.

ii. **Popularity:** SNMP is virtually supported by every enterprise network equipment manufacturer in the world. Its centralized management system is an extremely effective and widespread solution to network management. Because TCP/IP networks have become so popular, implementation and compatibility have become easy.

iii. **Efficiency:** SNMP also utilizes the User Datagram Protocol (UDP) to deliver packets called Protocol Data Units (PDUs). UDP is a quick method of transmitting data because it has low overhead costs. Unlike TCP, UDP lacks much of the acknowledgement features that guard against broken transmissions. Thus, the intermittent messages SNMP sends and the constant flow of status updates and alerts are kept at a minimum compared to TCP.

iv. **Easily Expandable to meet increased needs:** The control network administrators have with SNMP is extremely beneficial. With it, they are able to monitor and change network performance according to its needs. This proves vital with growing networks.

2.6.4 DISADVANTAGES OF SNMP

The drawbacks found in SNMP include the following:

i. **Simplicity:** Because SNMP uses UDP as its transmission protocol, it lacks many reliability and security issues. UDP runs on a very rudimentary level, using only the most basic transmission segments. While this connectionless protocol runs with fewer network resources, it does not ensure the data is correctly received. As networks increase in size, an increase in polling may be required to manage the system. This can increase the overhead of resources and would be inefficient.

ii. **Security:** Security has been a big concern with SNMPv1 and SNMPv2. Neither provides adequate security features such as management message authentication and encryption. With these holes in security, an unauthorized user could execute network management functions. Networks can be brought to a crawl if a malicious user carries out these actions. Deficiencies such as these have led many operations to have read-only capability. SNMPv3 addresses these issues and provides security enhancements in this area.

iii. **Network Traffic:** It generates a lot of network traffic as it polls devices for status information.

2.7 AGENT SYSTEMS

Currently, there exist no agreements on the concept definition of what an agent really is in the agent community and researchers have not been able to agree upon a common definition. However, agent is a software entity which functions continuously and autonomously in a particular environment, able to carry out activities in a feasible and intelligent manner that is representative to changes in the environment [21]. Another common definition was given by Peter and Stuart (2002), agent performs two tasks: It senses its environment through sensors and performs actions through effectors. An agent is a computational system that is long-lived, has goal, sensors, and effectors, and decides autonomously which actions to take in the current situation to maximize progress toward its time-varying goal [12].

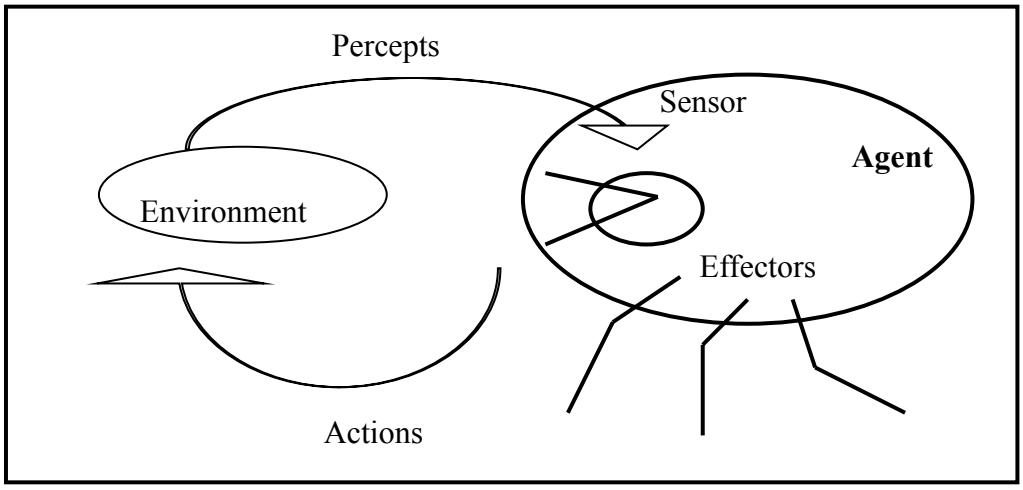

Figure 2.2: An agent in action

Usually, an agent that functions continuously in an environment over a long period of time would be able to learn from its experiences. An agent that inhabits an environment with other agents and processes to be able to communicate and cooperate with them, perhaps move from one place to place in doing so. In general, agents must be autonomous, able to execute without user intervention. They must be able to communicate with other software or human agents and to perceive the environment in which they reside.

Moreover, an agent in the content of software is essentially a self-contained software program module that is programmed to carry out certain actions on behalf of a human user or other software entity in a certain software environment. The agent can perform such things as searching for information, negotiating services, executing specified tasks, or collaborating with other agents. These actions are conducted in an autonomous fashion that requires little or no human intervention [5]. There are yet new achievements in agent-based systems constituting of the new domain of Distributed Artificial Intelligent (DAI). The agent-based technology has been employed to carrying a number of tasks including but not limited to production planning, scheduling and execution control, enterprise integration

and supply chain management as well as materials handling and inventory management [17].

However, in SNMP the managed device contains a software entity called an agent. In contrast, the SNMP defined agent does not qualify as an agent. The SNMP agent has absolutely no degree of autonomy or intelligence. The SNMP framework was written with the assumption that network devices have limited computing resources available, and therefore must rely on the NMS for instructions and computational processing. Even considering the SNMP Trap, the agent is limited to a fixed set of predefined thresholds that are statically implemented. This exemplifies the wide abuse of the term "agent" [5].

2.7.1 AGENT ATTRIBUTES

Software agent technology is at present employed as a promising way to support, and implement complex distributed systems and a useful supplement to client-server system [13]. A software agent is a software program that is capable of autonomous (or at least semi-autonomous) actions in pursuit a specific goal. Agents are needed because many of our everyday tasks are computer based and require the above mentioned properties for their execution. The following are some of the characteristics of an agent [6, 9]:

i. **Autonomy:** This is the ability of an agent to be able to operate without the direct intervention of humans. An autonomous agent is a system situated within and as a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda so as to effect what it senses in the future. An agent should be able to preemptive and independent actions on behave of user and that will be of benefit to the user.
ii. **Reactivity:** An agent should be able to perceive its environment and responds urgently by doing something when an event occurs.

iii. **Cooperation:** The agent is able to communicate and negotiate with other agents and/or their users; it is deliberative and may coordinate its actions with others. Collaborative agents are particularly useful when a task involves several systems on the network.

iv. **Mobility:** The agents can move within one environment or move from their environment to another environment and can carry data along with intelligent instruction that can be executed remotely.

v. **Learning:** This is the ability of an agent to acquire knowledge and use it to modify its behavior. Agent continuously adapt to changes in the environment.

vi. **Pro-activeness:** This is the ability of agents to generally follow plans, or at least execute rules when the environment reaches a known threshold by taking initiative.

vii. **Planning:** The agent organizes by priorities the actions to perform during its life. This one of the most important property of agents to possess.

viii. **Delegation:** This is the ability of an agent to ask another agent to perform one of its goals or tasks. This capacity is very important for balancing resources.

2.7.2 CLASSIFICATION OF AGENT

An agent is an intelligent system situated is some environment, and that is capable of autonomous action in the environment in other to meet its design objective [17]. An agent can be qualified as more intelligent, according to these categories [4, 11];

i. **Reflex Agent:** This is an agent that statically reacts to a percept according to some condition-action rule.

ii. **Reflex Agent with Internal State:** A state persists from one activation of the agent to the other and may be taken into account when firing rules.

iii. **Goal Based Agent:** This is an agent with an internal state, which does not only react, but also pursues an assigned goal.

iv. **Utility-Based Agent:** This is a goal based copping with several goals by ascending a certain utility to each state of its environment.

2.7.3 TYPES OF AGENTS

Agents are special software that involve in communications, bargaining, coordination, and perform so many other actions autonomously same as being done in real life. The application areas of agents technology and the combination of their different features results in different agent types [6, 9]. The following are some of the common types of agents:

i. **Collaborative Agents:** These are the agents that are able to emphasize autonomy and cooperation with other agents in order to perform their tasks for their users.
ii. **Mobile Agents:** These are the agents that can move from one computer to another computer and interact with their host and capable of roaming in a wide area networks.
iii. **Reactive Agents:** These are special agent that do not possess internal symbolic of their environments but they can act or respond in a stimulus-respond manner to the present state of the environment where they are embedded.
iv. **Information Agents:** These are the agents that can perform the role of managing, manipulate or gathering information from many distributed sources and use them to answer queries posed by other agents or their users.
v. **Intelligent Agents:** These are the agents that carry out some set of operations on behalf of a user or another program with some degree of independence. They act as autonomous personal assistant and work together with the users to accomplish some tasks.
vi. **Hybrid Agents:** These agents combine two or more agents' philosophy within a singular agent.

2.8 INTELLIGENT AGENTS

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. Autonomous in this sense means that agents are able to act without the intervention of humans or other systems: they have control both over their own internal state, and over their behaviour. A common view of an intelligent agent is that of an active object or bounded process with the ability to perceive, reason, and act. It is very important to note that the definitions above refer to the general concept of "agents" and not necessarily "intelligent agents".

Intelligent is general terms to describe some of mind activities such as judging, logical thinking, reasoning, planning and learning. Artificial Intelligence (AI) is based on intelligent behaviour, the principle of such intelligence is practically the same of human intelligence [10, 2]. To be intelligent, agents must be able to work together on solving problems in a dynamic environment and must be able to communicate understandable results back to the user. Intelligent agent is an agent that can adapt to the needs of different users, it can learn new concepts and techniques, it can anticipate the needs of the user, and it can take initiative and make suggestions to the user [21].

An intelligent agent can operate in real time and use natural language to communicate; and, it is able to learn from the environment and be adaptive to user behaviour. Different kinds of intelligent agents have different subsets of properties. Thus, the various disciplines have different views of an intelligent agent [5]. Intelligent agents exhibit the following characteristics: autonomy, social ability, reactivity, pro-activeness, mobility, learning, and beliefs. An intelligent agent is an independent entity capable of performing complex actions and resolving management problems on its own [15]. An unintelligent agent is distinguished from an intelligent agent based on the agent's properties [12].

2.8.1 COMMON USES OF INTELLIGENT AGENTS

Intelligent agents are now being used or are in development for a variety of uses. Since these software agents perform specific tasks as directed by the user, these programs are often called "bots" (robots). Intelligent agent software programs are in use for the following common functions [32]:

A Personal Applications

a. Information-Seeking "bots" [also called "infobots"; *weak-agency* ones "go fetch," *strong-agency* ones learn user's preferences and apply these to future searches].

- Search software - "electronic librarians" that search information databases and prioritize or rank selections
- Independent search tools - "data detective" software
- Data mining tools (information-based)

b. Shopping "bots" [also called "shopbots"]

- General on-line shopping -- "What's new and interesting?"
- Search-based shopping -- "I know what I want. Find it."
- "Bots" buying personal or business software ("softbots")
- "Bots" monitoring Web sources and updating you about items you have indicated you are interested (like about sales)

c. Personal Assistants

- Adapt searches (info or shopping) to user's personal preferences
- Keeps personal calendar (birthday reminders, meetings, daily events, etc.)

- Notifies user of updates of interest (info updates on news items of interest, or shopping items)
- Gathers data about your Website visitors, keeps a profile of their interests, keeps an e-mailing list so you can contact people when you have news they might be interested in)

B. **Business Applications**

a. Sales and Marketing

- Targeting customers (market-information data mining)
- Catalog sales offerings
- Helping customers (providing information about goods and services)

b. Banking

- Loan services for customers (seeking best interest rate, pre-qualifying procedures)
- Electronic money transfer (automates protocols and procedures)
- Tracking industry trends
- Application processing (loans)

c. Telecommunications

- "Smart phones" (operation or coordination of cellular phones, digital pagers, digital data displays)
- Human-computer interface (HCI) -- "talking heads" or "social interface" programs (may include some "natural language" use)
- Tracking, routing (anticipating or maximizing usage capacities)

d. Expert Systems (specialized knowledge-based applications)

- Medical use (diagnosis processing, treatment regimens, patient record/payment info-tracking, and insurance coverage processing)
- Law (case classification and analysis, precedent tracking)
- Law enforcement (suspect identification, case comparison, and info-tracking)
- Translation services (conversion of text written in one language into another language)

e. Computing and Network Systems Management

- Computer diagnostics, configuration, maintenance, and repair
- Network system diagnostics
- Networked program installation
- Network maintenance and repair
- E-mail filtering and sorting
- Network monitoring and protection programs

2.9 MULTI-AGENT SYSTEMS

An agent is a software component capable of flexible autonomous operation in a dynamic, unpredictable, and open environment [14]. When the single agent is unable to solve a problem because of its inadequate competence and the complexity of the problem, therefore several software agents from a distributed loosely coupled network and work together to solve the problem. A multi- agent system consists of several agents capable of reaching the goal collectively.

A Multi-agent system is a platform composed of multiple agents that interact to solve problems beyond their individual capabilities or knowledge. A Multi-Agent System (MAS) is a subset of Distributed Artificial Intelligence (DAI). DAI is concerned with problem-solving where agents solve tasks in a collaborative and cooperative manner, in a distributed environment [5, 2]. The concepts that are important and have to be taken into consideration in studying a multi agent system are the following as highlighted by Michaud [9, 10]:

- **i.** Consider what protocol the system uses for communication. Is it flexible, and does it provide directed communication and/or multicast communication?
- **ii.** Is the programming language a standard language, easy to use, compatible with other components, and portable?
- **iii.** Is the system flexible that can easily be adjusted to a particular application, and what are the constraints and requirements?
- **iv.** Is the system object oriented, layer based and well designed?
- **v.** Can the agents be visualized, does the user interact with the system, and is it easy to start using the system?
- **vi.** Are there any security features provided, and the communication among agents encrypted?
- **vii.** Are there any extra-features available, are the agents mobile, and are there any coordination constructs available?

Within a multi agent system, agents can communicate with each other using Agent Communication Languages (ACLs), which resemble human-like speech actions more than typical symbol-level program-to-program communication protocols [18].

2.9.1 ADVANTAGES OF MULTI-AGENTS

Agent-based technologies are considered the most promising means to deploy enterprise-wide and worldwide applications that often must operate across corporations and continents and inter-operate with other heterogeneous systems. Multi-Agent System has the following advantages over a single agent or centralized approach [2, 9, 10]:

a) Multi-agent system distributes computational resources and capabilities across a network of interconnected agents. Multi agent is decentralized and thus does not suffer from the "single point of failure" problem associated with centralized systems.

b) It allows the interconnection and interoperation of multiple existing legacy systems.

c) Multi-Agent System models solve problems in terms of autonomous interacting component-agents which is proving to be more natural way of representing task allocation, term planning, user preferences, open environment etc.

d) It provides solution in situations where expertise is spatially and temporally distributed.

e) It efficiently retrieves, filters, and globally coordinates information from sources that are spatially distributed.

f) Multi-Agent System enhances overall system performance, specifically along the dimensions of computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility, and reuse.

2.10 INTELLIGENT AGENTS IN NETWORK MANAGEMENT

Multi-agent systems are applied in the real world to graphical applications such as computer games. Agent systems have been used in films and also used for coordinated defense systems. Other applications include transportation, logistics, graphics, GIS as well as in many other fields. It is widely being advocated for use in networking management and mobile technologies, to achieve automatic and dynamic load balancing, high

scalability, and self-healing networks [20]. The following are arguments for intelligent agent in network management as found by group of researchers at Lucent Technologies, Inc. [5]:

1. Dynamism

The growing incompatibility of multi-vendor equipment and dynamic changes in network topologies has caused increased complexity and dramatic structural changes in network architectures. Intelligent Agent systems are better suited for managing such dynamically changing environments because agent-based systems handle dynamism in a natural way. Traditional and other management frameworks are inflexible and limited compared to agent-based systems. Frameworks based on SNMP and CMIP require software recompilation to handle changes, and offline time is required to activate new or modified software. CORBA-based systems implemented with solely static invocations suffer from the same problems. The use of Dynamic Invocation Interface (DII) with COBRA presents a time-consuming nuance relative to agent-based systems - it is cumbersome for programmers to code. Web-based solutions are completely reactive, which means that if the Network Management System doesn't ask for data then it will not be delivered.

2. Multiple Standards

The diverse user requirements for the various network technologies have led to the creation of several different network management standards and proprietary products. At the highest level, agents use a uniform communication means (e.g. KQML) for interaction between heterogeneous agents. Any other standard can be applied at the content level. No other network management technology has this degree of flexibility.

3. Interoperability

Network management interoperability issues have resulted from the development of dissimilar network management information models and manager-agent communication protocols. In agent- based systems the meta-layer for exchanging communications acts allows for coherent exchange of data at the content level. Agents know exactly what kind of data to expect and what its means.

4. Distribution

Current solutions for distributed management, such as RMON, are perhaps not truly distributed due to their inherent centralized nature. CORBA, as a distributed Network Management solution, suffers from many problems such as information bottlenecks and single points of failure. The advantage of an agent- based system is that it is inherently distributed. The communication language provides for natural collaboration between agents at various levels. This allows for a bottom-up approach to problem solving and utilization of available services at any level and stage. The mobility of agents is also an advantage. Mobility and intelligent distributed processing can provide efficiencies such as lower bandwidth requirements for network management.

5. Security

The agent-based framework does allow for security schemes at several layers. This framework is convenient to enforce security because agents decide at run time whether a received request will be fulfilled. All the other current technology frameworks provide security mechanisms because the industries mandate it within the standards. SNMP and CMIP standards mandate security schemes that have to be implemented at the design and deployment phases. CORBA provides security mechanisms integrated with the platform.

6. User Experience

As networks continue to expand, more and more users will become involved in some form of network management. Agent-based systems stand out in user experience because they

deliver plug-and-play networks. Network elements do not have to be provisioned because the agents residing on them can negotiate the conditions for incorporating them into the network. Similarly, services can be plugged into the network and made available automatically due to negotiation and directory services.

7. Rapid Software Delivery

Agent-based systems are very promising in rapid design, deployment, and maintenance of software products, as well as controlling the costs of expansions and modifications. They are designed with high-level concepts because the platform handles most low-level technicalities such as message passing through method invocation. Thus, systems can be designed by network management experts as opposed to programmers. In contrast, architectures built on top of SNMP and CMIP standards are the most expensive to design, implement, deploy, and maintain. Modifications require plenty of changes to the configuration data as well as off-line time.

8. Cost

Cost is a major factor for any organization when deciding to procure a software system especially when buying enterprise NM software. So much so that cost cutting has led to the evolution of management tools from proprietary and semi-proprietary solutions based on specialized platforms to standards-based solutions. Agent-based systems result in lower expenses for Network management. The value of the use of artificial intelligence techniques in providing human-like behavior will result in substantial savings in direct costs due to the decreased requirement for human operators. This holds true for the other technologies, but the capabilities of agent-based systems provide a much greater value. The evaluation findings provide comparative proof and demonstrate the potential for

intelligent-agent-based technologies as an overall better solution for enterprise NM, relative to SNMP and the other common protocols.

CHAPTER THREE
ANALYSIS OF THE EXISTING SYSTEM

3.1 INVESTIGATION OF THE EXISTING PLATFORMS

Platform is software components that can range from a collection of classes compatible among them to a high-level programming environment equipped with several tools helping programming agents. The purpose of a platform was to ease the task as much as possible. Several works have been done on the intelligent multi-agent technologies for network management and many people are currently working on such systems. The major aim of the search is to find a platform which includes multi-agent features and some kind of intelligent. In other word, the platform had to possess some reasoning abilities as well as provide the features that will permit users to set up a distributed multi- agent. The main task was to search for a single platform encompassing several properties. Therefore the properties of such a platform revolve around three issues stated below:

i. Platform must allow easy implementation of the intelligent agents. This simply means that the agent must be Autonomy, Reactivity and has the capability of inferring actions from given goals or the capability of planning their actions.

ii. The platform must also provide an infrastructure for agents, which should include:
- High-level of communication between agents, as message passing or as reactivity to events generated by other agents.
- Easy creation and management of a distributed population of agents.

iii. The platform had to be freely available (and availability of its source code). This is to better understand its functioning and also to modify it.

However, it was discovered that most platforms did not satisfy all the criteria. The properties was then reduced and divided into three parts:

i. Infrastructures that will provide the means to create distribute and execute agents.
ii. Intelligence of an inference engine or planner.
iii. Communication that will provide high-level communication language, like KQML classes.

3.2 MULTI-AGENT SYSTEM DESIGN TOOLS

There is need for a multi agent platform in order to perform network management with intelligent agents [9]. There are various design tools available for the design of multi-agent systems. The development of these design tools ranges from commercially available products to research prototypes. Some tools provide graphical interfaces for the development of the multi-agent systems, and several tools do not provide a graphical design environment. There are two types of the agent platforms; one is for implementing and deployment of multi agent systems which includes JADE, FIPA-OS, Zeus, etc. The other is for the execution and roaming of the agents and their security on the network such as TACOMA, Telescript, and Aglets etc. A lot of platforms were found during the investigation. Possibly some other platforms may have not been come across which might be better than the ones discussed below. Although some platforms were not investigated at all because they are commercial product or the code is not freely available or they are not implemented in Java. The following were found during the investigation of the existing platforms [6, 9, 31]:

1. JATLite

JATLite stands for Java Agent Template Lite. It was developed by the Computer Science Department at Stanford University [9]. JATLite is a set of Java packages that make it easy to build multi-agent systems using Java. It is a package of Java classes and programs that allow users to create quickly new systems of software agents that communicate over the Internet in order to perform a distributed computation. It provides basic communication

tools and templates for developing agents that exchange KQML messages through Internet standards, *TCP/IP,* SMTP, and FTP. It provides code for creating agents and provides a robust agent infrastructure, in which agents register with an Agent Message Router (AMR), using a name and password, in order to be able to exchange buffered messages with and transfer files between other agents on the Internet (some of which may be Java applets), and connect/disconnect/reconnect from/to the joint computation [9, 28].

When an agent is created and connected to the network, it first registers with the ANS (Agent Name Server) which store all the names and the addresses of existing agents. By registering, the agent passes the ANS its name, port number and the domain of its local host. There is no requirement for installation of special software to host agents and no special host is assumed for any agent. However, JATLite does provide essential functionality required for building a multi-agent application; it does not define a methodology for specifying the social behavior of agents. Moreover, the concepts of the ANS and the Agent Router are inherently centralized in nature. Each time an agent joins the system, it has to register with ANS; and when it leaves the system, then the ANS also has to be informed. All communication must go through the Agent Router. Thus, any application developed using JATLite cannot be truly scalable.

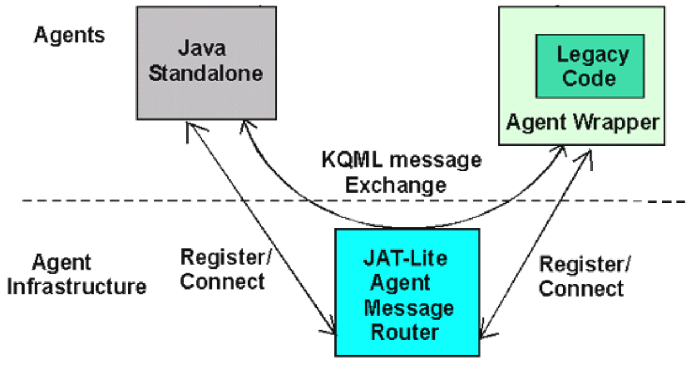

Figure 3.1: JATLite Agent Message Router [6, 9]

2. Aglets

Aglet is a framework for Java "Agent Applets" designed by IBM Japan. An aglet is a Java object that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its state (data). A built-in security mechanism makes it safe for a computer to host un-trusted aglets [6, 9, 28]. Aglets Workbench is a visual environment for building network based applications that use mobile agents to search for, access and manage corporate data and other information. IBM Aglets are mobile Java programs which may travel and execute in specialized nodes in the network. The Java Aglet Application Programming Interface of the framework defines the methods necessary for Aglet creation, message handling in the network and initialization, dispatching, retraction, deactivation/activation, cloning and disposing of the Aglet [9, 1].

The Aglets workbench includes an Agent Web Launcher named Fiji and a Visual Agent Manager named Tahiti. Fiji is a Java applet based on the Aglets Framework and therefore capable of creating an Aglet and retracting an existing Aglet into a client's web browser. Tahiti uses a unique graphical user interface to monitor and control Aglets executing on a given computer. It also implements a configurable security manager that provides a fairly high degree of security for the hosting computer system and its owner. Although, Aglet is more intended to allow agents to move than a framework for multi-agents, it can be combined with JKQML to allow communication among agents. JKQML was developed to provide a framework and API for constructing Java-based, KQML speaking software agents that communicate over the Internet [6, 9].

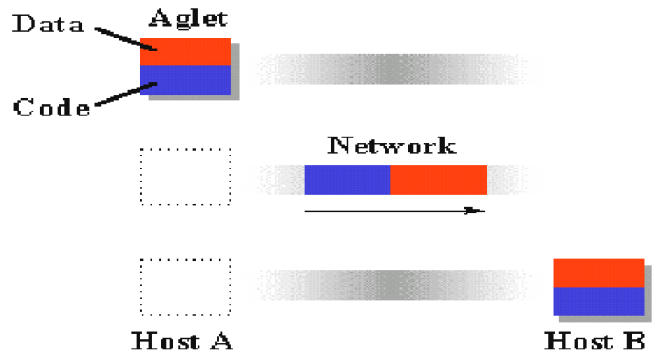

Figure 3.2: Aglets Serialization through the network [6, 9]

Based on KQML, JKQML has been developed by IBM to provide a framework and API for constructing Java-based, KQML-speaking software agents that communicate over the Internet. JKQML allows the exchange of information and services between software systems, creating loosely coupled distributed systems. JKQML provides flexibility for the extension of the framework, and it supports the following three protocols [9]:

i. **KTP (KQML Transfer protocol):** A socket-based transport protocol for a KQML message represented in ASCII.
ii. **ATP (Agent Transfer Protocol):** A protocol for KQML messages transferred by a mobile agent that is implemented by Aglets.
iii. **OTP (Object Transfer Protocol):** A transfer protocol for Java objects that are contained in a KQML message.

3. Concordia

Concordia was developed by Mitsubishi Electric Information Technology Center of America. It is a full featured Java based framework for development and management of network-efficient mobile agent applications for accessing information anytime, anywhere, and on any device supporting Java [9]. Concordia offers a flexible scheme for dynamic

invocation of arbitrary method entry points within a common agent application. It provides support for agent persistence and recovery and guarantees the transmission of agents across' a network. Concordia was designed to provide fairly complete security coverage from the outset [6, 9].

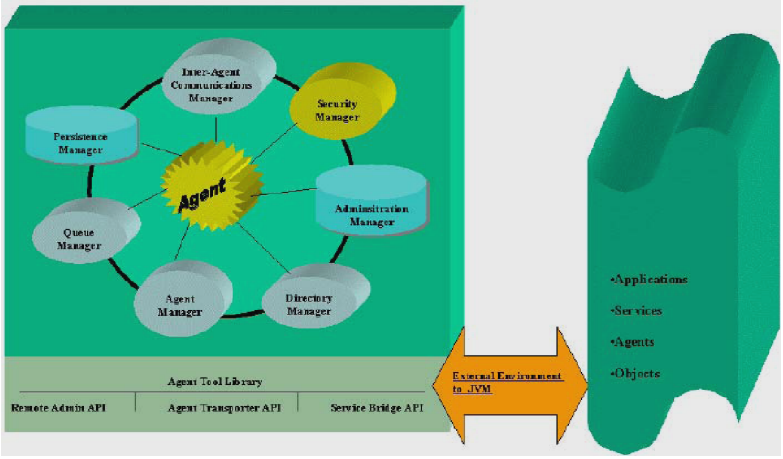

Figure 3.3: Concordia sewer architecture [6, 9]

Although Concordia provides a useful set of services for implementing agent mobility, security, persistence and transmission, it does not provide any methodology to specify how agents in a multi- agent system coordinate cooperate and negotiate to bring about a coherent solution. Emphasis here is on the communication aspect in an agent based application. Moreover, the fact that the agent Itinerary is outside the agent implies that where the agent travels is maintained in a separate logical location regarding the place where the agent lives. However, the results in Concordia agents are not being totally autonomous [9, 30].

4. Odyssey

Odyssey is an outgrowth of Telescript, the first mobile agent commercially available. It is General Magic's implementation of mobile agents in Java. Odyssey is an agent system

implemented as set Java class libraries that provide support for developing distributed mobile applications. Odyssey technology implements the concepts of places and agents. It models a network of computers, however large, as a collection of places. A place offers a service to the mobile agents that enter it. A communicating application is modeled as a collection of agents. Each agent occupies a particular place. However, an agent can move from one place to another, thus occupying different places at different times. Agents are independent in that their procedures are performed concurrently. Odyssey provides Java classes for mobile agents and stationary places [6,9].

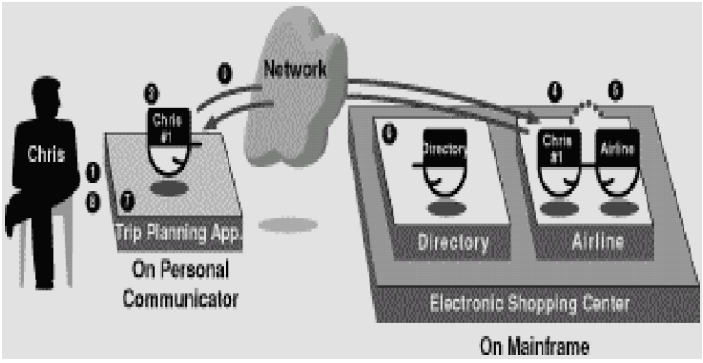

Figure 3.4: A typical Odyssey Application [6, 9]

The current version of Odyssey provides the basic functionality of mobile agents but it does not go beyond. The developer must adhere to a very rigid structure while implementing mobile applications. Unlike IBM Aglets and Concordia, it does not provide an extensive security mechanism. Odyssey does not support broadcast communication and speech-act messaging. Presently on the downhill slope, Odyssey has been passed by its concurrent mentioned above. Nonetheless, the director of IBM's Aglets program has recently joined the General Magic's Odyssey team, so there might be a future convergence between the concepts of these two tools [9].

5. Voyager

Voyager was developed by ObjectSpace Inc as a Java-based agent-enhanced Object Request Broker (ORB). It combines the power of mobile autonomous agents and remote method invocation with complete CORBA support and comes complete with distributed services such as directory, persistence, and publish subscribe multicast. It allows Java programmers to quickly and easily create sophisticated network applications using both traditional and agent-enhanced distributed programming techniques. Voyager agents roam a network and continue to execute as they move. Voyager can remotely construct and communicate with any Java class, even third party libraries, without source. It allows seamless support for object mobility. Once created, any serializable object can be moved to a new location, even while the object is receiving messages. Messages sent to the old location are automatically forwarded to the new location.

Voyager uses regular Java message syntax to construct remote objects, send those messages, and move them between applications. Voyager allows agents (i.e. autonomous objects) to move themselves and continue executing as they move. In this way, agents can act independently on behalf of a client, even if the client is disconnected or unavailable. This approach is particularly valuable in any type of workflow or resource automation. Voyager is a very efficient tool for constructing agent-based distributed applications. However, it does not provide any classes for defining the social behavior of agents, does not support broadcast communication and speech-act messaging, and lacks in security [6, 9].

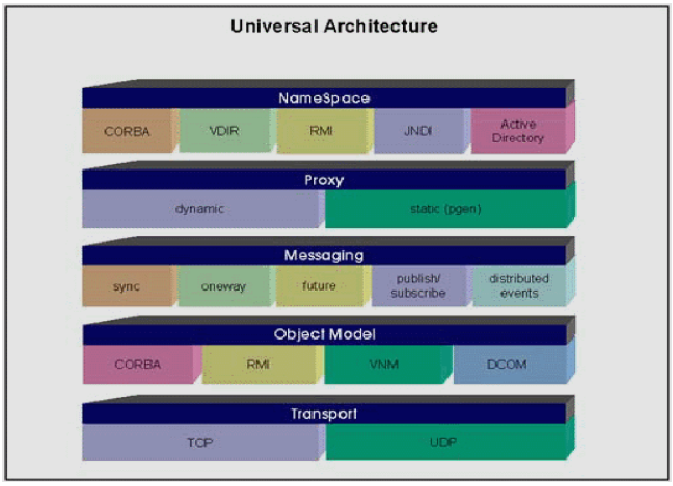

Figure 3.5: Voyager Universal Architecture [6, 9]

6. IA Factory

IA Factory was developed by Bits and Pixels Company of America [9]. The goal of the IA Factory is to supply the programmer with an Application Program Interface (API) that precludes him from having to go through the entire network programming and debugging. This framework provides a generic agent that allows a programmer to extend its specific behavior. In the simplest case a table of behavior is sufficient. When an agent requires greater complexity, programmers can extend a class to give an agent complex and interesting behavior [6, 9].

The Intelligent Agent Factory reduces the time needed to create intelligent agent. The agents are intelligent in the sense that they are controlled with rules written in Jess (CLIPS), a forward-chaining system. Agents and rules are generated from simple specifications of workflows [28]. The IA Factory creates a set of agents, along with an interface to run them. These agents communicate via the KQML language (there is also another version of the library for commercial use that uses XML messages). The agents are lightweight, which means that hundreds of agents can run within one Java Virtual machine.

The source code to the agents is generated in Java; hence, they can be customized to increase the functionality of intelligent agents.

7. RETSINA

RETSINA stands for Reusable Environment for Task Structured Intelligent Network Agents. It is an open multi-agent system (MAS) that supports communities of heterogeneous agents developed at Carnegie Mellon University [9]**.** The RETSINA system has been implemented on the premise that agents in a system should form a community of peers that engage in peer to peer interactions. Any coordination structure in the community of agents should emerge from the relations between agents, rather than as a result of the imposed constraints of the infrastructure itself. In accordance with this premise, RETSINA does not employ centralized control within the MAS; rather, it implements distributed infrastructural services that facilitate the interactions between agents, as opposed to managing them.

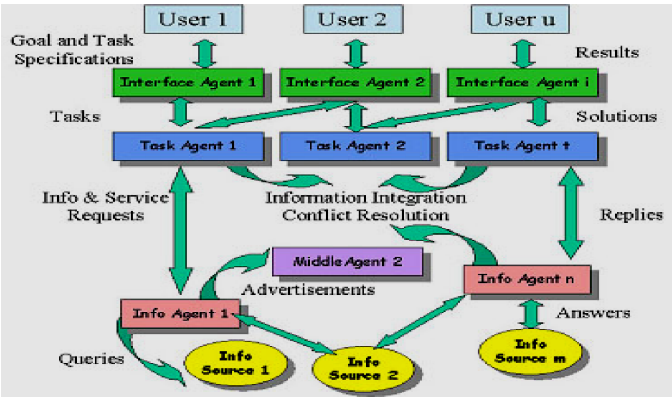

Figure 3.6: Retsina Multi Agent System [6, 9]

The RETSINA framework is being used to develop distributed collections of intelligent software agents that cooperate asynchronously to perform goal-directed information retrieval and information integration in support of performing a variety of decision-making

tasks. A collection of RETSINA agents forms an open society of reusable agents that self-organize and cooperate in response to task requirements. Their designer focused on three crucial characteristics of the overall framework that differentiate RETSINA from others [6, 9]:

- Use of a multi-agent system where the agents operate asynchronously and collaborate with each other and their users.
- Agents actively seek out information.
- Information gathering is seamlessly integrated with problem solving and decision support.

The RETSINA functional architecture consists of four basic agent types:

a. **Interface agents:** Interact with users, receive user input, and display results.

b. **Task agents:** Help users perform tasks, formulate problem-solving plans and carry out these plans by coordinating and exchanging information with other software agents.

c. **Information agents**: It provides intelligent access to a heterogeneous collection of information sources.

d. **Middle agents:** It helps match agents that request services with agents that provide services.

RETSINA addresses the problem of how to facilitate communication among agents of different types. As part of the RETSINA infrastructure of reusable agents, middle agents represent an important step in our ongoing effort to provide a foundation that will allow heterogeneous agent types and architectures to interoperate successfully. Each RETSINA agent has four reusable modules for communicating, planning, scheduling, and monitoring the execution of tasks and requests from other agents.

- The Communication and Coordination module accepts and interprets messages and requests from other agents.
- The Planning module takes as input a set of goals and produces a plan that satisfies the goals.
- The Scheduling module uses the task structure created by the planning module to order the tasks.
- The Execution module monitors this process and ensures that actions are carried out in accordance with computational and other constraints. Below is a graphic representation of the RETSINA agent architecture:

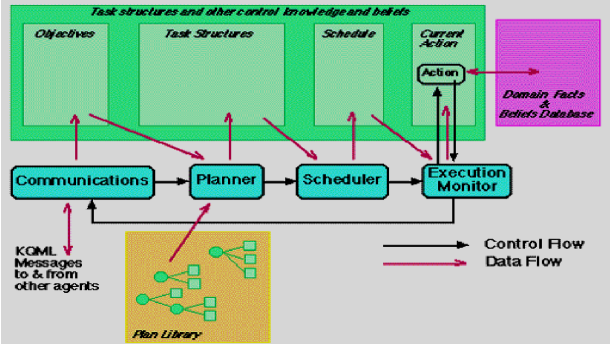

Figure 3.7: RETSINA agent architecture [6, 9]

The set of Java classes within RESTINA emphasized on how agents can find information or advertise their capabilities. No particular attention is given regarding the behavior of each agent and how they interact with other agents. The Name Server API uses a centralized approach, making the system less scalable and fault-tolerant. As RETSINA is an open system, any agent on the Internet can communicate and interact with the actual community [6, 9]. It seems like RESTINA is not the adequate platform for the need of network management system [6].

8. MAST

MAST (for Multi-Agent System Tool) was developed by the Intelligent Systems Group of the Department of Telemetric Systems Engineering at the Technical University of Madrid. It is a heterogeneous, multi-agent, general purpose MAS. It employs a decentralized model of control and consists of two basic types of entities: agents and the network. *Agents,* defined as autonomous entities that way carry out specific tasks by themselves or in conjunction with other entities, and *the network* through which they interact. MAST agents consist of a structured set of elements that include services, goals, resources, internal objects, and control. Control in MAST is merely a specification of how an agent handles a service request. The network consists of a yellow page service incorporated in an agent that facilitates lookup services. MAST is a loosely coupled system or agents that use the Common Knowledge Representation Language (CKRL) to communicate and MAST-ADL (Agent Description Language) to describe agents [15]. The MAST architecture is a very complete multi-agent system tool. However, the fact that it is implemented in C++ makes it less portable and less multifunctional than Java-based frameworks. Many of the services within MAST (interoperability between heterogeneous agents) are unnecessary because they are already included in the Java language [6, 9].

9. dMARS

dMARS is an agent-oriented development and implementation environment designed for building complex, distributed, time-critical systems. It is intended for rapid configuration and ease of integration, and it helps with system design, maintenance, and reengineering. dMARS agents are designed according to the BDI (Beliefs, Desires, and Intentions) model. They are able to reason about their environment, their beliefs, their goals, and their intentions. They model their expertise as a set of context-sensitive plans. These plans can both react to changes in the environment and proactively pursue the agent's objectives.

However, dMARS multi-agent systems can be implemented as lightweight processes within a single UNIX process, as separate UNIX processes on the same machine, or as a distributed configuration communicating over a TCP/IP network. Interfacing with other processes is achieved via a simple, well-defined communication protocol. The system provides comprehensive libraries and components to support the development, implementation and testing of an application, therefore minimizing the need to develop application-specific support code [6, 9]. dMARS provides a high-level, graphical plan language for specifying complex, context-dependent processes and tactics. dMARS is written in C/C++, thus, does not provide for true architecture neutrality and portability. Unlike Java-based systems, applications developed in dMARS can only run on limited platforms and require using different compilers for different platforms. It supports only a limited number of C++ compilers [6, 9].

10. JAFMAS

JAFMAS stands for Java-based Framework for Multi-Agent Systems [28]. JAFMAS provides a methodology and a framework for designing multi-agent system. JAFMAS is a methodology and architecture for the design and implementation of multi-agent systems in Java. It focuses on communication, agent interaction, and multi-agent system coherency and coordination. It removes the difficult and time-consuming task of manually writing code to support communication between agents in a multi-agent system.

JAFMAS uses the five-step methodology [28]:

- Identify the Agents
- Identify the Conversations
- Identify the Conversation Rules

• Analyze the Conversation Model

• Implementation of Multi-agent system

JAFMAS software architecture provides a set of classes for the implementation of the multi- agent system, as proscribed by the methodology. These classes provide support for the sending and processing of broadcast and directed communication, conversation execution, and system coordination. JAFMAS allows the designer to step through the methodology using a set of graphical user interfaces, with the result being a set of generated class files that are compiled into a multi-agent system [27]. Each agent in a JAFMAS-based multi-agent system can have a list of 'Required Resources.' These 'Required Resources' establish the needs of the agents in the system, and allow for the discovery of agents that provide that resource. When the multi-agent system is started the agents use multi-cast messages to find the other agents in the system that provide their 'Required Resources' [6, 9, 31].

11. Microsoft Agents

Microsoft Agent is a set of programmable software services that supports the presentation of interactive animated characters within the Microsoft Windows interface. Developers can use characters as interactive assistants to introduce, guide, entertain, or otherwise enhance their Web pages or applications in addition to the conventional use of windows, menus, and controls. Microsoft Agent enables software developers and Web authors to incorporate a new form of user interaction, known as conversational interfaces, that leverages natural aspects of human social communication. In addition to mouse and keyboard input, Microsoft Agent includes optional support for speech recognition so applications can respond to voice commands. Characters can respond using synthesized speech, recorded audio, or text in a cartoon word balloon.

The conversational interface approach facilitated by the Microsoft Agent services does hot replace conventional graphical user interface (GUI) design. Instead, character interaction can be easily blended with the conventional interface components such as windows, menus, and controls to extend and enhance your application's interface. Microsoft Agent's programming interfaces make it easy to animate a character to respond to user input. Animated characters appear in their own window, providing maximum flexibility for where they can be displayed on the screen. Microsoft Agent includes an ActiveX control that makes its services accessible to programming languages that support ActiveX, including Web scripting languages such as Visual Basic Scripting Edition (VBScript). This means that character interaction can be programmed from HTML pages. Microsoft Agent requires Microsoft Windows, Internet Explorer and ActiveX [31].

12. AgentBuilder

AgentBuilder is a commercial product produced by Reticular Systems, Inc. It provides an integrated toolkit for the construction of intelligent agents. AgentBuilder provides a fully featured set of graphical interfaces for the design and development of a multi-agent system. It provides various graphical tools including project management, ontology management, agent management, protocol management, and run time engines. AgentBuilder is Java-based and its communication language is KQML. The AgentBuilder agent model consists of agents that have beliefs, capabilities, commitments, intentions; and behavioral rules. Given an agent model's initial state and external influences the agent operates and communicates with the external world, which in turn modifies its states [Reticular 1999]. An interesting agent based system designed using AgentBuilder is a prototype system for buying and selling power generation capacity [31].

13 ADE

Another commercial multi-agent design tool is ADE (Agent Development Environment) by Gensym. ADE is an Integrated Development Environment for Distributed Multi-Agent Applications, which is built on G2, an Object-Oriented graphical environment. ADE provides a graphical language called AdeGrafcet (an extension of the French Grafcet standard, which is based on Petri net representations) for the specification of the agent's behavior. ADE provides a communication middleware, utilizing subject-based addressing. ADE also has a simulation environment for simulating the design before implementation, as well as a debugging environment [31].

14. ZEUS

ZEUS is a toolkit for building distributed multi-agent systems. ZEUS specifies a design approach utilizing a three-layer model of an agent, the definition layer, the organization layer, and the coordination layer. It provides a set of graphical interfaces to allow a designer to create the multi-agent system. An agent is comprised of libraries that include communication (using KQML), social interaction, multi-agent coordination, and planning and scheduling. This tool generates source code for the agent-based system in Java [31].

15. DECAF

DECAF (Distributed, Environment-Centered Agent Framework) is a software toolkit for developing intelligent agents. It provides user interfaces for the specification of the agent behavior, including logical and temporal constructs. It allows reuse of generic agent behaviors. Its Plan- Editor removes the designer of a multi-agent system from the low level API and allows for the rapid development of agents, which are generated in Java. DECAF also provides some design time error checking, and supports the use of agent communication languages [31].

16. MASSYVE Kit

The MASSYVE (Multi-agent Agile Manufacturing Scheduling Systems for Virtual Enterprise) kit is a software kit for the development of multi-agent systems. It allows a user to configure, modify, and create a multi-agent system; it provides user interfaces to guide a designer through the process. It also provides mechanisms to launch the agents, both locally and remotely [31].

17. JADE

JADE (Java Agent DEvelopment Framework) is a software framework for developing FlPAcompliant multi-agent systems. JADE provides a set of interfaces for the design of agents, implemented in Java. JADE uses the FIPA-ACL, utilizing a combination of socket, RMI and CORBA. This framework also provides a set of pre-defined behaviors for the use of them designer, including 'Simple Behaviour', 'Cyclical Behaviour', and 'Non Deterministic Behaviour' [31].

18. KAoS

KAoS is an agent architecture that provides a form of agent-oriented programming. A set of structures: knowledge, desires, intentions, and capabilities represent the agents in KAoS. State transition diagrams specify the conversation policies [28].

19. Bond

Bond is a Java-based, object-oriented middleware for collaborative network agents. It presents a set of objects to enable agent communication, including Message Passing, Object Replication, and Directory Services. The bond agents communicate using KQML [28]. The tables below provide a survey of currently agent construction tools.

Table 3.1: Commercial Products of Agent Builder Tools [31]

S/n	Product	Assigned To	Language	Description
1.	AgentBuilder®	Saikumar Thota	Java	Integrated Agent and Agency Development Environment.
2.	AgentTalk	NTT	LISP	Multi-agent Coordination Protocol.
3.	Agentx	Daqing Wang	Java	Agent Development Environment
4.	Aglets	RAMIRO ARMIJOS	Java	Mobile agents
5.	Concordia	PavanKumar Gourisetty	Java	Mobile agents
6.	DirectIA SDK	MASA Group	C++	Adaptive Agents
7.	Gossip	Vasubabu	Java	Mobile agents
8.	Grasshopper	IKV++	Java	Mobile agents
9.	iGEN™	Derick Aranha	C/C++	Cognitive Agent Tootkit
10.	Intelligent Agent Factory	Raghu Ram Devineni	Java	Agent Development Tool
11.	Intelligent Agent Library	As above	Java	Agent Library
12.	JACK Intelligent Agents	Xiaoming Liu	JACK Agent Language	Agent Development Environment
13.	JAM	Hariprsad N Davanagere	Java	Agent Architecture
14.	Jumping Beans	Ad Astra Engineering, Inc.	Java	Mobile Components
15.	LiveAgent	Alcatel	Java	Internet Agent Construction
16.	MadKit	Wang, Lei	Java, Scheme, jess	Multi-agent Development Tool
17.	Microsoft Agent	Sheng Gao	Active X	Interface creature
18.	NetStepper	JW's Softwate Gems		Agent Development Environment
19	Network Query Language	Sudeep Suresh		Programming Language
20.	Pathwaker	Min Yue	Java	Agent-oriented Programming library
21.	Swarm	Rama Kanth Davarakonda	Objective C, Java	Multi-agent Simulation

22.	Topia Personal Agents	Topia Ventures		Mobile Agents
23.	UMPRS	Intelligent Reasoning Systems	C++	Agent Architecture
24.	Via: Versatile Intelligent Agents	Kinetoscope	Java	Agent Building Blocks
25.	Voyager	Paul Grosch	Java	Agent-Enhanced ORB

Table 3.2: Research Agent Builder Tools [31]

S/n	Product	Assigned To	Language	Description
1.	Agent Builder Shell (ABS)	Madhan Kumar	COOdination language	Agent Architecture
2.	Agent Factory	Sandeep Hulsandra	Smalltalk-80	Agent Prototyping Environment
3.	D'agents	Prashanth Talked		Retrieval, Organization, and Presentation
4.	Agent Tcl	Vaibhav Dani	Tcl	Mobile Agents
5.	Architecture type-based Development Environment (ADE)	University of Potsdam dept. of Computer Science Professorship of Software Engineering TAXT	Java	Platform and Application Independent Agent Development Environment.
6.	Ascape	The Brookings Institution	Java	Software Framework
7.	Bee-gent	Fu-Sheng Lu	Java	Software Development Framework
8.	Bond Distributed Object System	Jianfeng Tang	Java	Agent Framework
9.	Cable	Logica Corporation	Agent Devination Language, C++	System Architecture
10.	DECAF Agent Framework	Srikanth Balusani	Java	Agent Framework
11.	dMARS	Australian Artificial Intelligence Institute Ltd.	C, C++	Agent-oriented Development and Implementation Environment
12.	EXCALIBUR	Xu, Chunxiang		Autonomous Agent Architecture in a

				computer Game Environment
13	JADE	TILAB, formally CSELT	Java	Multi-Agent Development Framework

3.3 CRITERIA FOR CHOICE OF PLATFORM

There is need to consider some practical criteria that one has to take into account when choosing a platform. In particular, these criteria can condition the adoption of a platform to a particular project:

- **Availability:** Is there a trial version, confidential clauses, is the source code available? How much does it cost?
- **Support:** What are the future developments of this platform? Is this platform used in large scale?

Based on the study of the existing platforms, java-based platforms become the most serious contender because of the following:

a. Java Virtual Machine that interprets Java byte codes, Java applications are greatly potable and their behaviour varies little from one type of platform to the other. Portability is needed when working in heterogeneous environments like the ones involved in network management: nodes of a network are actually likely to be running different operating systems.

b. Java is such a language that is strongly object-oriented and it enjoys many publicly available libraries and applications. This is because different products needed to be

highly and easily inter-compatible. Thus, the language had to be an object-oriented one to facilitate integration and a widespread one.

3.4 CHOICE OF THE PLATFORMS

The following were considered for the choice of the platform:

i. To be popular and regularly maintained (for bug fixes and extra features),
ii. To be grounded on well-known academic models,
iii. To be developed with industry-like quality standards,
iv. To cover as many aspects as possible of Multi-Agent Systems, including agent models, interaction, coordination, organization, etc.,
v. To be simple to set-up and to evaluate. This includes good documentation, download availability, simple installation procedure, and multi-platform support.

However, there is need to deliberately avoid platforms:

i. That are still in experimental state, abandoned, or confidentially distributed,
ii. That are grounded on non-existent models,
iii. That covers only one aspect of multi-agent systems, like single agent platforms, mobile agent platforms, interaction infrastructures toolkits,
iv. That they are too short on some construction stages, like purely methodological models or development tools without methodology.

3.5 THE PROPOSED PLATFORM (JADE)

Java Agent Development Environment (JADE) is a software framework to aid the development of agent applications in compliance with the Foundation for Intelligent Physical Agents (FIPA) for inter-operable intelligent multi-agent systems [8]. JADE complies with FIPA, which includes the Agent Management System (AMS), the default Directory Facilitator (DF) and the Agent Communication Channel (ACC). JADE

automatically activates these three agents when the agent platform starts-up. [8, 28, 29]. The goal of JADE is to simplify the development while ensuring standard compliance through a comprehensive set of system services and agents. JADE can be considered as an agent middle-ware that implements Agent Platform and a development framework. It deals with all those aspects that are not peculiar of the agent internals and that are independent of the applications, such as message transport, encoding and parsing, or agent lifecycle [29]. The platform provided by JADE is distributed. It can be split over several hosts with one of them acting as a front end for management and inter-platform communication [8]. A JADE platform comprises one or more agent containers, each living in one Java Virtual Machine (JVM) and providing the execution environment for the agents. There is one Main Container acting as the front end. It has the supervisory control over the JADE platform. The AMS, the DF and the ACC are in the Main Container.

3.6 REASONS FOR THE CHOICE OF JADE

JADE can arguably be considered the most popular software agent platform available today, however, the following are the reasons why JADE was the final choice:

i. JADE is particularly suited to create application that run on distributed low cost architectures and perfectly meets the scalability requirement. Scalability was certainly the main reason that leads to widely usage of JADE.

ii. It has the capability to serve well both reactive processes, such as activation and troubleshooting, and proactive processes.

iii. JADE has the capability of executing long and fairly complex tasks such as network element configurations upload (considering response times and network latency).

iv. It has the capability of automatically exposing agent capabilities as Web Services by means of WSIG (Web Services Integration gateway) add-on.

v. JADE agent oriented approach intrinsically facilitates the isolation of functional modules (i.e. agent) that can be implemented separately.

vi. JADE allows seamlessly spreading components (i.e. agents) across different hosts thus guaranteeing a high degree of scalability.

3.7 JADE DESCRIPTION

Java Agent Development Environment (JADE) is a software framework that makes easy the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi agent systems [29, 30]. FIPA (Foundation for Intelligent Physical Agent) is an international non-profit association sharing the effort to produce specifications for generic agent technologies. JADE is an open source project that has a goal of simplifying agent development and ensuring standard compliance through a comprehensive set of system services and agents. JADE is a software framework to facilitate the development of interoperable intelligent multi agent system that is used by a heterogeneous community of users as a tool for both supporting research activities and building real applications [30]. JADE can be thought of three important features:

i. JADE is a runtime system for FIPA-compliance multi-agent systems, supporting application agents whenever they need to exploit some feature covered by the FIPA standard specifications such as message passing, agent life cycle management etc.

ii. JADE provides a proper set of functionalities to simplify development of multi agent systems but it puts very few restrictions on the user code without in particular imposing any specific agent architecture.

iii. JADE can be deployed both JEE, JSE, and JME devices and its runtime provides a homogenous set of APIs that independent of the underlying network and Java technology.

JADE can be considered as middle-ware that implements an efficient agent platform and supports the development of multi agent systems. It tries to keep high performance of a distributed agent system implemented with the Java programming language [29]. JADE can arguably be considered the most popular software agent platform available today. Several companies have used and continue to use it in the context of very different application domains, including network management, supply chain management, rescue management, fleet management, health care, auctions, tourism, etc. For instance, British Telecommunication's Intelligent Systems Research Centre uses JADE as the core platform of mPower, a multi agent system that is used by BT engineers to support cooperation between mobile workers and team-based job management, i.e. teams rather than individuals receive job assignments and team members must cooperatively manage the assigned jobs. Several universities use JADE, exploiting its Open Source availability and its technical excellence, both to study and teach the strength of the software agent technology and to extend its functionalities with new add-ons [30].

3.8 JADE ARCHITECTURE

JADE provides a distributed platform that can be split over several hosts with one of them acting as a front end for management and inter-platform communication [31]. Its communication architecture tries to offer flexible and efficient messaging, transparently choosing the best transport available and leveraging state-of-the-art distributed object technology embedded within Java runtime environment [29, 30, 31].

The JADE platform architecture is firmly rooted in peer-to-peer ideas and communication patterns. JADE is fully developed in Java and, from a software engineering standpoint, values highly, among others. JADE platform comprises one or more agent containers, each living in one Java Virtual Machine (JVM) and providing the execution environment for the

agents. There is one main container acting as front end. It has the supervising control over the JADE platform. The Agent Management System (AMS), the Directory Facilitator (DF) and the Agent Communication Channel (ACC) are in the main container. JADE automatically activates these three agents when the platform start-up. The figure below is the JADE platform general architecture.

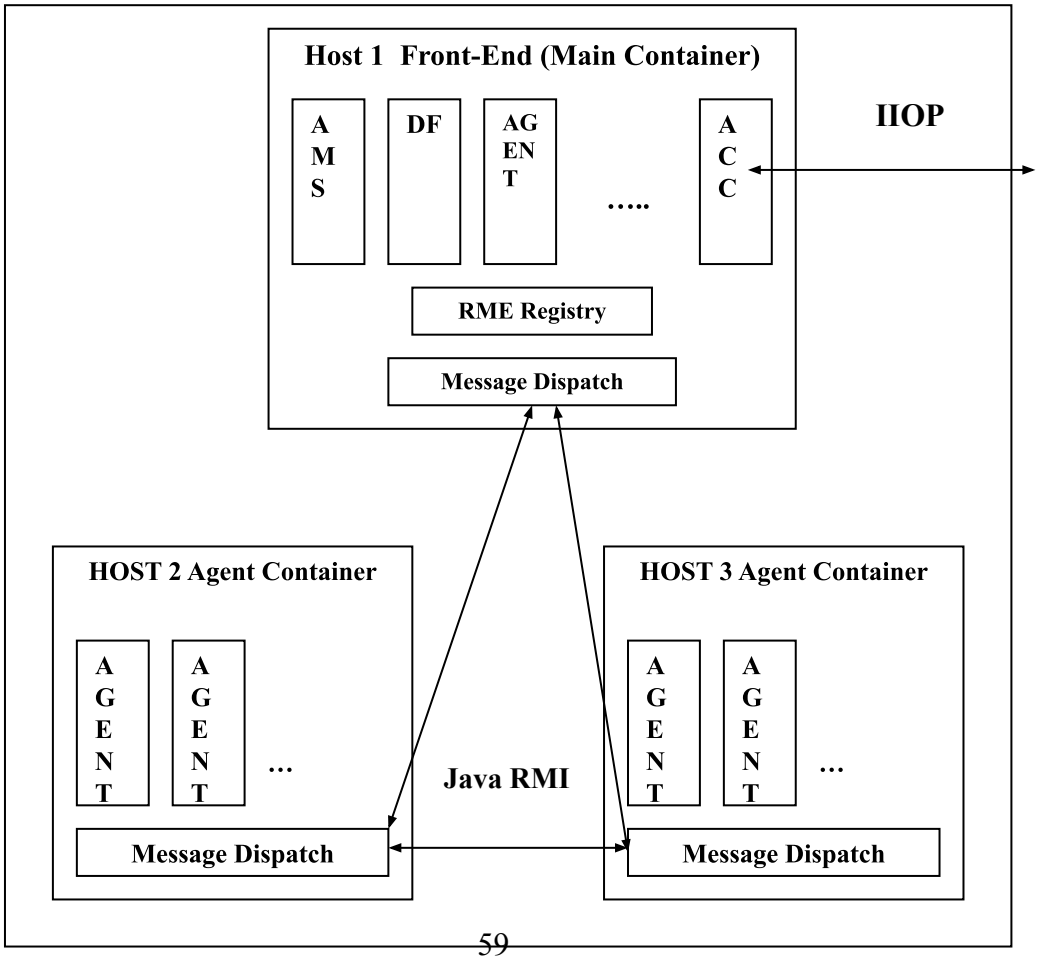

Figure 3.8: Software Architecture of a JADE Agent Platform

JADE platform includes both libraries (i.e. the Java class) required to develop application agents, and the runtime environment that provides the basic services and that must be active on the device before agents can be executed. Each instance of the JADE run-time is called container (because it “contains” agents). The set of all containers is called platform and provides a homogenous layer that hides to agents (and to application developer as well) the complexity and diversity of underlying tiers (hardware, operating system, type of network, JVM).

However, JADE design choice (in particular not to commit too strongly to specific agent architectures) and its modular architecture allowed JADE to fit the constraints of environments with limited resources. JADE has also been integrated into complex server-side infrastructures such as .NET, and JEE where JADE becomes a service to execute multi-party proactive applications. An application based on JADE usually made of a set of components referred to as agents. Each agent have unique name, they execute tasks and communicate by exchanging messaging. These agents live on top of platforms which usually provide them with basic services such as message delivery. Platform composed of one or more containers and those containers can be executed on different hosts, thus achieving distributed platform. Each container can contain zero or more agents. There is a special container called main container in the platform. This main container is the first container to start in the platform and other containers register to it at bootstrap time. It also contains two special agents called the AMS (Agent Management System) and DF (Directory Facilitator). The figure below is a typical architecture of JADE architecture:

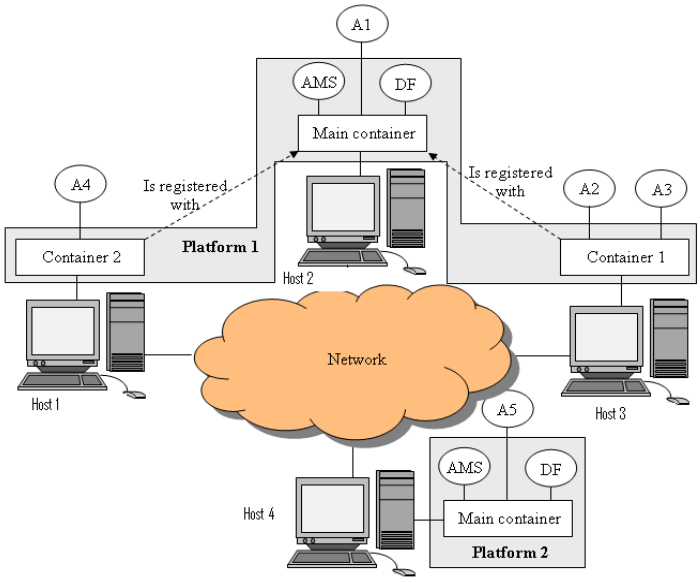

Figure 3.9: The architecture of JADE

3.9 FEATURES OF JADE

JADE is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required. JADE is completely implemented in Java language and the minimal system requirement is the version 1.4 of JAVA (the run time environment or the JDK).

The goal of JADE is to ease development while ensuring standard compliance through a comprehensive set of system services and agents. To achieve such a goal, JADE offers the following features to the agent programmer:

1. *FIPA-compliant Agent Platform*, including the *AMS*, the *ACC* and the default *DF* mandatory system agents. All these three agents are automatically activated at platform startup.
2. *Distributed Agent Platform*. The agent platform can be split on several hosts and only one Java Virtual Machine is executed on each node. Agents are implemented as Java threads and a suitable transport is chosen for message delivery, depending upon relative location of sender and receiver agents.
3. *Multiple Domains support*. A number of *FIPA* compliant *DF* agents can be started at runtime and linked in a federation, thereby implementing a multiple domain agent environment.
4. *Multithreaded execution environment with two-level scheduling*. Every JADE agent runs within it own thread of control, but also is able to run multiple *behaviours* concurrently. A preemptive scheduling is performed among all agents within a single Java Virtual Machine, whereas cooperative scheduling is used for the various task of a single agent.
5. *Object Oriented programming environment*. Most concepts present in *FIPA* specifications are represented as Java classes, so that a uniform programming interface is presented to users. For example, *ACL* messages and fipa-agent-management ontology objects all have a suitable Java counterpart.
6. *Library of interaction protocols*. Ready to use behaviour objects are provided for the standard interaction protocols such as fipa-request and fipa-contract-net. To build an agent that can act according to an interaction protocol, application developers just need to implement domain specific actions, while all application independent protocol logic will be carried out by JADE framework.
7. *Administration GUI*. Common platform management operations can be performed through a graphical user interface, showing active agents and agent containers.

Using this *GUI*, platform administrators can create, destroy, suspend and resume agents, besides creating domain hierarchies with multiple federated *DF* agents.

3.10 AGENT COMMUNICATION

Agents communicate between themselves by the use of message passing. Agents can communicate transparently regardless of whether they live in the same container, in different container (in the same or in different hosts) belonging to the same platform or in different platforms. It uses the FIPA-ACL language to represent messages. JADE tries to choose the most efficient way to pass message between agents as the message protocol, this is to minimize the communication overhead. The communication paradigm adopted is the asynchronous message passing.

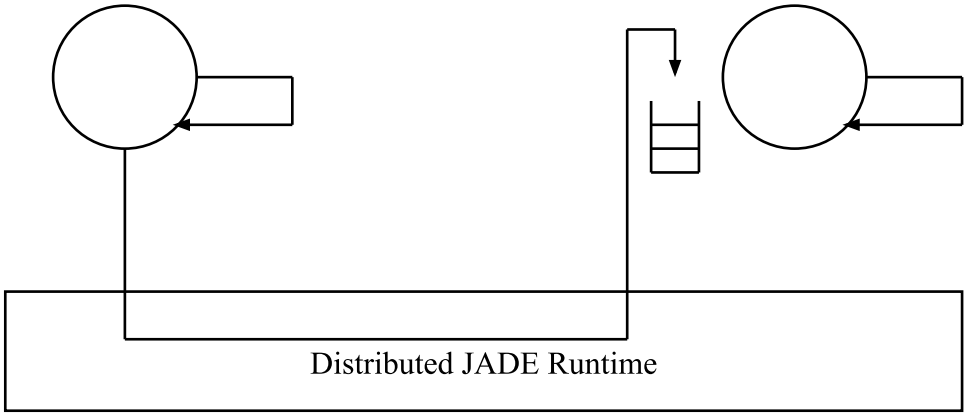

Figure 3.10: *The JADE asynchronous message passing paradigm*

Each agent has a sort of mailbox (the agent message queue) where the JADE runtime posts messages sent by other agents. Whenever a message is posted in the message queue the receiving agent is notified. If and when the agent actually picks up the message from the message queue to process it is completely up to the programmer.

JADE uses three ways to pass messages [8]:

1. **Receiver in the same container of the same platform:** Java events are used; the cost is a cloning of the ACLMessage object and a local method call.
2. **Receiver in a different container of the same platform:** Java RMI is used; the cost is a message serialization on the sender side, a remote method call and a message serialization on the receiver side.
3. **Receiver on a different platform:** CORBA IIOP is used; the cost is the conversion of the ACLMessage object into a string object and an IIOP marshalling on the sender side, a remote method call and an IIOP un-marshalling followed by ACL parsing on the receiver side.

Communication is based on asynchronous message passing paradigm. Message format is defined by the ACL language defined by FIPA. An ACL message contains a number of fields including the sender, the receiver, the communication act, and the content. Inter-platform communication (i.e. communication between agents living on different platforms) is based on modules called Message Transport Protocol (MTP). Such modules are able to marshal/un-marshal and transmit ACL messages according to the FIPA specifications. In this way, JADE agents are able to communicate with agents living on remote platforms regardless of whether these are other JADE platforms or different platforms provided that they are FIPA compliant. FIPA specifies how to transfer ACL messages over three well known transport protocols: HTTP, IIOP (the transport protocol defined in CORBA) and SMTP. JADE provides suitable MTPs for HTTP and IIOP only.

CHAPTER FOUR
SYSTEM DESIGN AND IMPLEMENTATION

4.1 JADE PROGRAMMING MODEL

JADE is fully object-oriented, Like all of the other popular programming languages used to create database-driven software. JADE was designed to have all the most important features of object-oriented programming, but it was also designed to make programming simple, and so does not exhibit the full range of tools that some other languages do. For example, JADE does not support the overloading of methods or operators. This may seem like a big loss to some programmers, but for programming database applications which is what JADE is designed for, the parts that are left out do not end up being major drawbacks, as they are almost never needed. One notable feature that JADE lacks is parameterized constructors and this lead to some dangerous consequences in that one can never know if an object has been properly initialized.

From a fault tolerance standpoint, JADE does not perform very well due to single point of failure represented by the Front End container and, in particular, by the AMS [15]. Classes in JADE are kept together in schemas. Schemas serve the same purpose as Java packages or namespaces in .NET, but they are much different in the fact that schemas have a hierarchy, and inherit classes from super schemas. This becomes useful especially when programming using the model-view-controller methodology, as model classes can be put in one schema, and then the controller and view classes can be built on top of the model classes in a subschema.

4.2 JADE PROGRAM STRUCTURE

JADE applications are structured quite differently from most programming languages in the fact that JADE programs are *not* developed by writing code into long files and then

compiling all the files together at once. JADE programs are actually developed using a user interface that allows programmers to visually create classes and define their properties and methods. Instead of locating methods in large files, programmers select the method they would like to edit and only the code for that particular method is displayed. Also instead of compiling all the code of a program at once, in JADE, each method is compiled individually as soon as the method is completed, meaning code can be checked immediately.

All the code for a JADE application is stored in its object-oriented database. This has several advantages. First, it allows for multi-user development, as the database maintains concurrency control. Second, with each piece of the code being a separate object in the database, in a lot of the cases it is possible to recode the system while it is live and online as long as the parts of the system being changed are not in use.

4.3 OBJECT MODEL

JADE is an object-oriented software development and deployment platform. It has its own programming language that exhibits a seamlessly integrated application server and object database management system. It is designed to be an end-to-end development environment, which allows systems to be coded in one language from the database server down to the clients. There are four classes needed to be inherited and extended, these are:

i. The ‘CreateAgent’ class provides a graphical interface to let the user enters the parameters to an agent.
ii. The ‘Agent’ class implements the local behaviour of the agent.
iii. The ‘Conversation’ class is extended for each conversation that the agent can have.
iv. The ‘ConvRule’ class is extended for each rule changing the state of conversation.

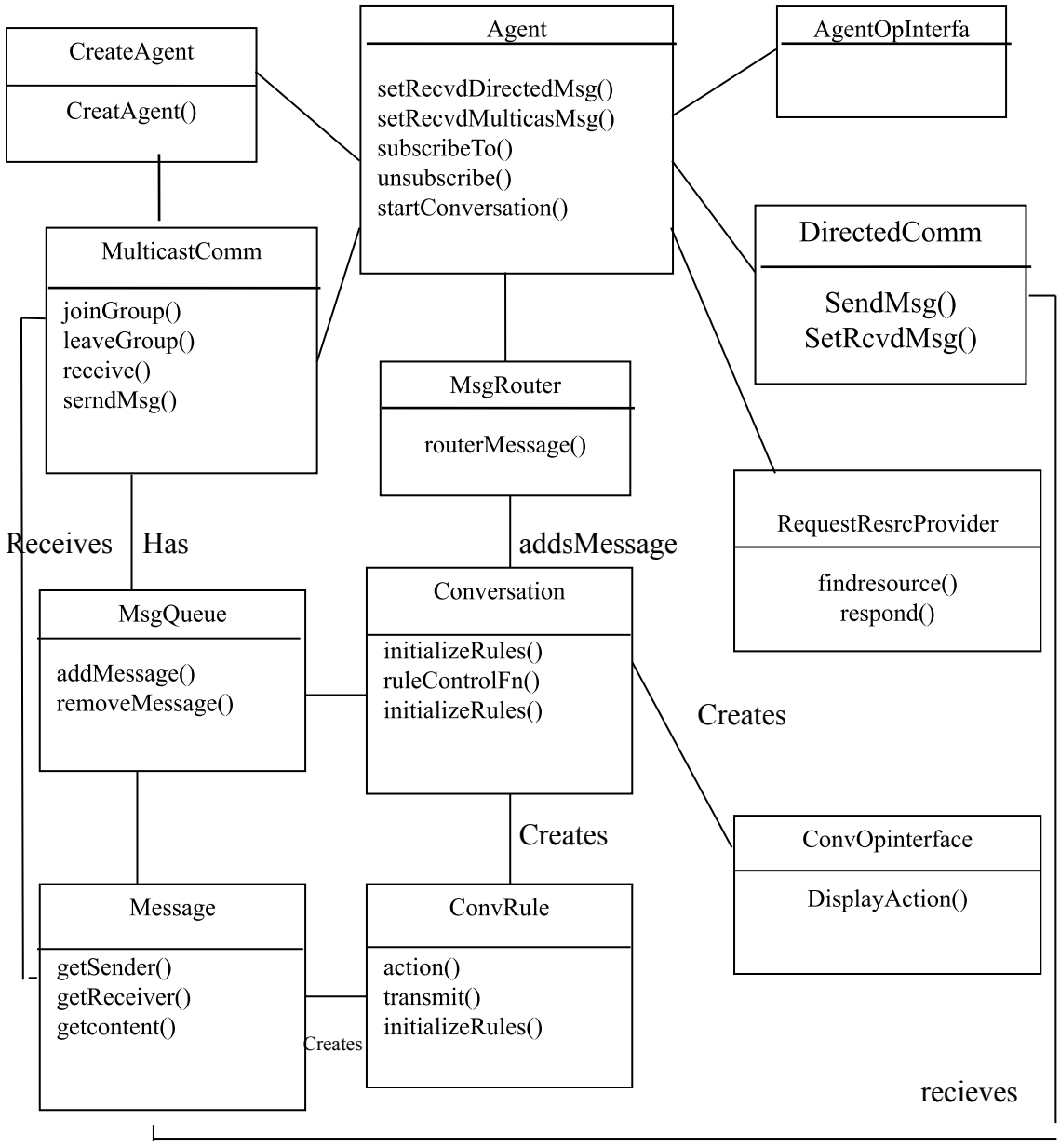

Figure 4.1: Object Model Design [9]

4.4 OBJECT DATABASE

The most difference between JADE and other object-oriented programming languages is that its object database is a native part of its language. For example, when creating an object in JADE, it can be created as transient or persistent. Creating an object as **transient** is just the same as creating objects in other object-oriented programming languages. The object is simply created in memory, and then lost when the program ends. On the other hand, when an object is created as **persistent**, when the program ends, the object will still exist and be there the next time the program starts up. In other words, when an object is persistent JADE automatically works in the background to store and retrieve the object in the database when necessary. Persistent objects can be distributed across multiple co-operating servers, with JADE automatically handling object caching and cache coherency.

There are very few differences between manipulating transient and persistent objects. It has been said that JADE makes it *appear* to the programmer as if all the objects in the entire database were in local memory. Most of the time, JADE's object-oriented database is used in a multi-user system, and so this statement could be extended to say that JADE makes it appear to the programmer as if all the objects in the database were stored in some shared memory that all users connected to the system could access, even from different computers. With all of the program code centralised on the database server as well the data, JADE achieves its goal of an end-to-end system, as JADE presents such a level of abstraction that all client nodes can be programmed as if they were running on the database server. This is very desirable to most database programmers as they don't have to take several different technologies and link them together; they just create one application for everything.

JADE's database is also inherently object-oriented, and so it eliminates the performance loss in an object-relational mapping system where objects must constantly be converted

from object-oriented form to relational form. Like all other commercial database products, JADE is ACID-compliant and has all of the standard features such as atomic transactions, locking, rollback, crash recovery and the ability to keep one or more secondary database servers synchronized with the main database for backup, disaster recovery and performance reasons. JADE also provides a Relational Population Service that enables automatically replicating objects from the main database to one or more relational databases. This allows JADE systems to interoperate with relational databases for reporting, business intelligence and data warehouse purposes.

4.5 MAS IMPLEMENTATION

Usually, the network implementation has been included inside the agents. Therefore two main actors constitute the system: the router agent (including the router implementation) and the end-host agent. This system is organized as pictured model below,

4.5.1 ROUTER AGENT

The router agent is an agent managing a network node. It is composed of two layers:

a. Router Layer

The router layer is the network itself. Its goal is to route the arriving messages according to their destination. A simple routing table is associated to it so that the router knows where to forward the messages. A fixed size buffer is also used to temporary stock the arriving messages. The router treats the messages as fast as it can. If the buffer is overloaded, the extra messages are discarded. It is the work of the agent to synchronize the traffic so that no messages are lost. Each router possesses also a certain bandwidth. That means that no more bandwidth than what is possessed by the router can be allocated to user requests. A simple graphical user interface is provided so that information can be observed on what is happening. One can also change the router bandwidth, option useful for testing and debugging.

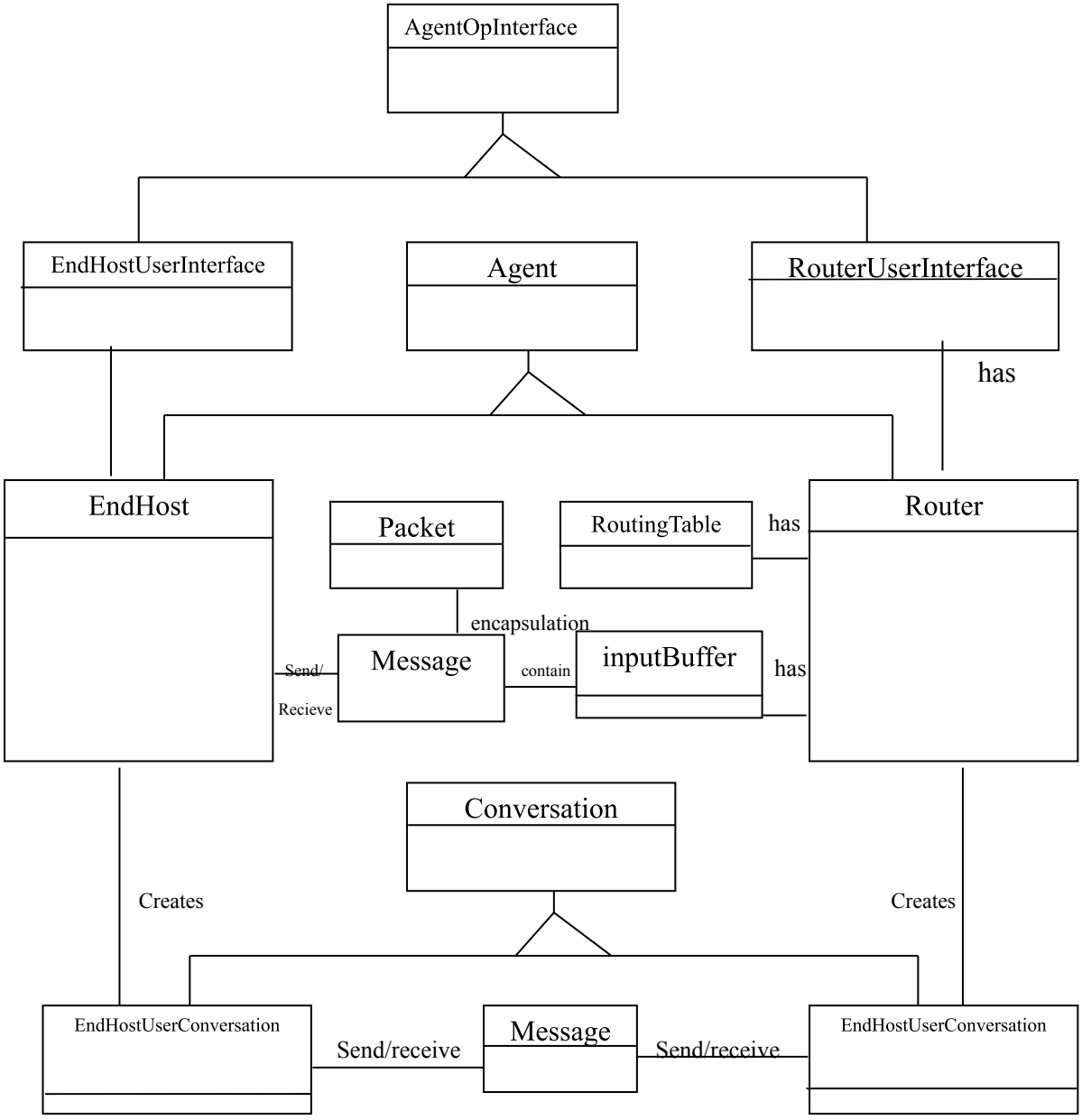

Figure 4.2: Overall MAS Implementation of agent [6]

b. Intelligent Layer

The intelligent layer is the actual intelligent agent. It communicates with other agent's intelligent layers and with the router layer in order to manage the node properly. When it has to accomplish a goal that involves other agents, it creates a Conversation object to talk with other agents and reach the given objective.

4.5.2 END-HOST AGENT

The end-host Agent plays the role of intermediate between the user and the network. A graphical interface is displayed so that the user can interact with the system. Ideally, the endhost agent should interface the multimedia application, so that the whole agent infrastructure is invisible to the user. The user has to explicitly reserve some bandwidth or send messages. Only fake data is sent, however the simulated network is able of transporting real data and therefore act as a real network. However, the end-host agent is not only a composition of graphical component. The underlying layer should support an intelligent infrastructure able of conversing with the network. An end-host is usually attached to a unique router. The end-host agent will have to talk with his router agent in order to reserve some bandwidth. It should also be able to interpret the demand requested by the user. For example, if the user asks for a good connection, the end-host agent should know (or learn with feedback from the user) that a good connection is about 500kb/s.

4.5.3 MESSAGE CLASS

The message class encapsulates all the information exchanged amongst agents. Every instance of that class has a performative variable. This variable represents the nature of interaction that is going to take place. The class provides methods for specifying both the sender and the receiver agent. It is also possible to specify the intent of the message. The intent enables the sender agents to express the intention which required them to send the

message, and the receiver agents to filter the message upon looking at only the intent slot. It also facilitates message routing. Messages are of two general types:

- **Declarative messages** are used to announce the presence of an agent. This type of message is used to set up the network at the beginning: each agent broadcasts his name and then waits for the agent he is connected to answer.
- **Content messages** contain a description of the piece of knowledge being accompanied with the actual data. They are used to transport data packets along the network. A data packet can be assimilated to an IP packet on the Internet. It possesses the sender address, the destination address and the actual data.

4.6 DESIGN METHODOLOGY

Five-step methodology was proposed for building a multi-agent application. The methodology focuses on logical issues of the problem being considered and then converges to the implementation. The stages are:

i. **Identifying the Agents:** Starting with the list of aims we want the system to satisfy, then considering various entities that will be interacting with each other to achieve this global aim. These entities represent the agents in the system. After identifying the agents, clarify the aims of each agent and the service they provide. The agents are divided into different categories by grouping the agents with similar aims and services into a single category. These categories help in identifying the agent classes in the application. After identifying the agent classes, then identify the application specific classes that each agent class may use.

ii. **Identifying the Conversations:** When developing agents, the world is being viewed as composing of intelligent things. At this stage, we begin to model the interaction between these things in the form of conversations. Identify every possible conversation an agent can engage in, and represent those conversations by

developing an automata model for each of them. These different automata models identify the different conversation classes in the application.

iii. **Identifying the Conversation Rules:** Each conversation is represented by an automata model. Alternative action of an agent produce different states in the conversation (automation) and the current state of a conversation influences how the agent react in the next moment. Conversations are rule-based descriptions of what agent does in certain situations.

iv. **Analyzing the Conversation Model:** It is important to do an analysis of the logical consistency of all agent conversation in order to design multi-agent system that finds a coherent solution to the entire system problem. They should be analyzed to verify the coherency of the system.

v. **MAS Implementation:** At this final stage, suitable tools for multi-agent application implementation will be chosen, this ensures communication, interoperation and coordination support.

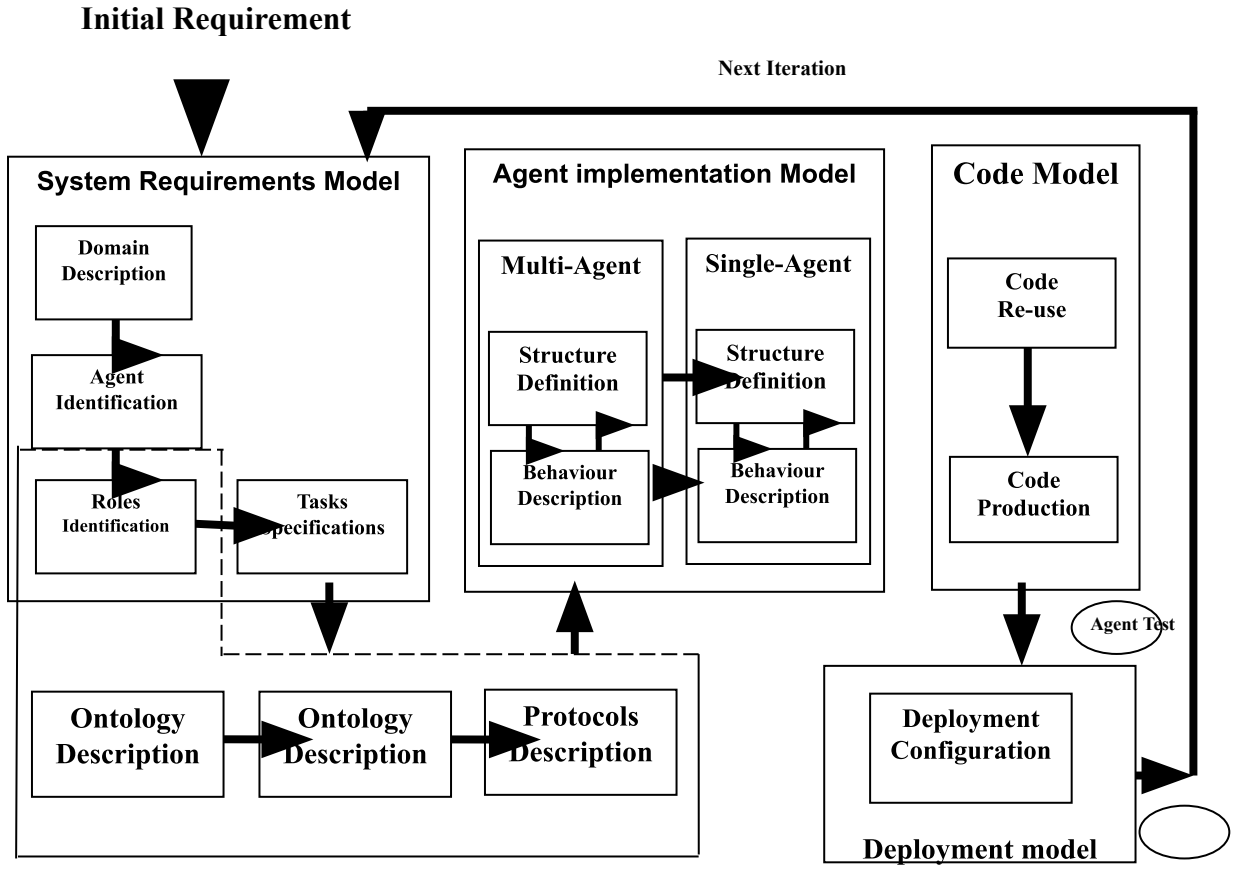

Figure 4.3: The Models for Design Methodology

4.7 CHOICE OF PROGRAMMING LANGUAGE

JADE is entirely written using the *Java* programming language, exploiting such advanced features as *Java RMI*, *Java CORBA IDL*, *Java Serialization* and *Java Reflection API*. The following are reasons why Java was use to design JADE:

i. Architecture Neutral and Portable

Because the agents are inherently distributed, applications must be able to execute anywhere on the network without prior knowledge of the target hardware and software platform. Java provides the advantages of architecture neutrality and portability to agent developers. If the Java run-time platform is made available for a given hardware and software environment, an application written in Java can then execute in that environment without the need to perform any special porting work for that application. The primary benefit of the interpreted byte code approach is that compiled Java language programs are portable to any system on which the Java interpreter and run-time system have been implemented.

ii. Multithreaded

The Java library provides a Thread class that supports a rich collection of methods to start a thread, run a thread, stop a thread, and check on the status of a thread. Java has built-in that supports threads which is one of the most powerful tools in Java, not only to improve interactive performance of graphical applications, but also to run multiple events concurrently. Multithreading is the way to obtain fast, lightweight concurrency within a single process space.

iii. Distributed

Java especially lends itself as an extremely suitable choice in this regard. It offers an extensive library of classes and routines which cope easily with both UDP and TCP/IP protocols, and thus supports sending both broadcast and directed messages across the network. In any MAS, agents residing on different machines or different environments need to communicate information and knowledge about their goals, beliefs and intentions to each other in order to coordinate and cooperate so as to bring about a coherent solution. Communication is a very important aspect in the development of any MAS.

iv. Secure

Java enables the construction of virus free, tamper free systems. Java is a strongly typed language and it does not support pointers, which make it a very robust language. The Java run-time system uses a byte code verification process to ensure that code loaded over the network does not violate any Java language restrictions. The authentication techniques are based on public key encryption.

v. Object Oriented

Java adopts the four principles of object oriented languages: inheritance, encapsulation, abstraction and message passing communication. Object oriented design is a very powerful concept because it facilitates the clean definition of interfaces and makes it possible to provide reusable software. The user has no choice but to encapsulate all data in objects. Since agents are essentially built around a group of different object components.

vi. Database Connectivity JDBC

The Java Database Connectivity kit lets Java programmers connect to any relational database, query it, or update it using the industry standard query language (SQL). This is a very useful feature as databases are among the most common uses of software and hardware today, and any application using Java can easily integrate with preexisting databases to update the local model of its agents. This is even more useful for MAS as agents usually possess a knowledge base. As this knowledge base gets bigger, it has to be organized and stored efficiently.

4.8 IMPLEMENTING JADE AGENTS

JADE is a middleware that enables fast and reliable implementation of multi-agent distributed systems and which can be integrated with Artificial Intelligence (AI) tools. Beyond a runtime library, JADE offers some tools to manage the running agent platform and to monitor and debug agent societies.

4.8.1 THE AGENT CLASS

To create a JADE agent is as simple as defining a class extending the jade.core.Agent class and implementing the setup() method as shown in the code below.

```
import jade.core.Agent;
public class BookBuyerAgent extends Agent {
                protected void setup() {
        // Printout a welcome message
System.out.println("Hello! Buyer-agent "+getAID().getName()+" is ready.");
}
        }
```

The setup() method is intended to include agent initializations. The actual job an agent has to do is typically carried out within "behaviours".

4.8.2 AGENT IDENTIFIERS

Each agent is identified by an "agent identifier" represented as an instance of the jade.core.AID class. The getAID() method of the Agent class allows retrieving the agent identifier. An AID object includes a globally unique name plus a number of addresses. The name in JADE has the form *<thename>@<platform-name>* so that an agent called *Ayour* living on a platform called *P1* will have *Ayour@P1* as globally unique name. The addresses included in the AID are the addresses of the platform the agent lives in. These addresses are only used when an agent needs to communicate with another agent living on a different platform.

Knowing the nickname of an agent, its AID can be obtained as follows:

```
String nickname = "Ayour";
AID id = new AID(nickname, AID.ISLOCALNAME);
```

The ISLOCALNAME constant indicates that the first parameter represents the nickname (local to the platform) and not the globally unique name of the agent.

4.8.3 RUNNING AGENTS

The created agent can be compiled as follows.

javac –classpath <JADE-classes> BookBuyerAgent.java

In order to execute the compiled agent the JADE runtime must be started and a nickname for the agent to run must be chosen:

java –classpath <JADE-classes>;. jade.Boot buyer:BookBuyerAgent

4.8.4 AGENT TERMINATION

Even if it does not have anything else to do after printing the welcome message, the agent is still running. In order to make it terminate its doDelete() method must be called. Similarly to the setup() method that is invoked by the JADE runtime as soon as an agent starts and is intended to include agent initializations, the takeDown() method is invoked just before an agent terminates and is intended to include agent clean-up operations.

4.8.5 PASSING ARGUMENTS TO AN AGENT

Agents may get start-up arguments specified on the command line. These arguments can be retrieved, as an array of Object, by means of the getArguments() method of the Agent class. The BookBuyerAgent can be given title of the book to buy as a command line argument. This can be done as follows

```
        import jade.core.Agent;
    import jade.core.AID;
public class BookBuyerAgent extends Agent {
    // The title of the book to buy
```

```
        private String targetBookTitle;
        // The list of known seller agents
        private AID[] sellerAgents = {new AID("seller1", AID.ISLOCALNAME),
new AID("seller2", AID.ISLOCALNAME)};
// Put agent initializations here
        protected void setup() {
// Printout a welcome message
        System.out.println("Hello! Buyer-agent "+getAID().getName()+" is ready.");
// Get the title of the book to buy as a start-up argument
Object[] args = getArguments();
        if (args != null && args.length > 0) {
targetBookTitle = (String) args[0];
System.out.println("Trying to buy "+targetBookTitle);
}
        else {
// Make the agent terminate immediately
System.out.println("No book title specified");
doDelete();
}
        }
// Put agent clean-up operations here
protected void takeDown() {
// Printout a dismissal message
System.out.println("Buyer-agent "+getAID().getName()+" terminating.");
}
        }
```

4.9 AGENT TASKS - THE AGENT BEHAVIOUR CLASS

The actual job an agent has to do is typically carried out within “behaviours”. A behaviour represents a task that an agent can carry out and is implemented as an object of a class that extends jade.core.behaviours.Behaviour. In order to make an agent execute the task implemented by a behaviour object it is sufficient to add the behaviour to the agent by means of the addBehaviour() method of the Agent class. The behaviours can be added at any time: when an agent starts (in the setup()method) or from within other behaviours.

Each class extending Behaviour must implement the action() method, that actually defines the operations to be performed when the behaviour is in execution and the done() method (returns a Boolean value), that specifies whether or not a behaviour has completed and have to be removed from the pool of behaviours an agent is carrying out.

4.9.1 BEHAVIOURS SCHEDULING AND EXECUTION

An agent can execute several behaviours concurrently. However it is important to notice that scheduling of behaviours in an agent is not pre-emptive but cooperative. This means that when a behaviour is scheduled for execution its action() method is called and runs until it returns. Therefore it is the programmer who defines when an agent switches from the execution of a behaviour to the execution of the next one.

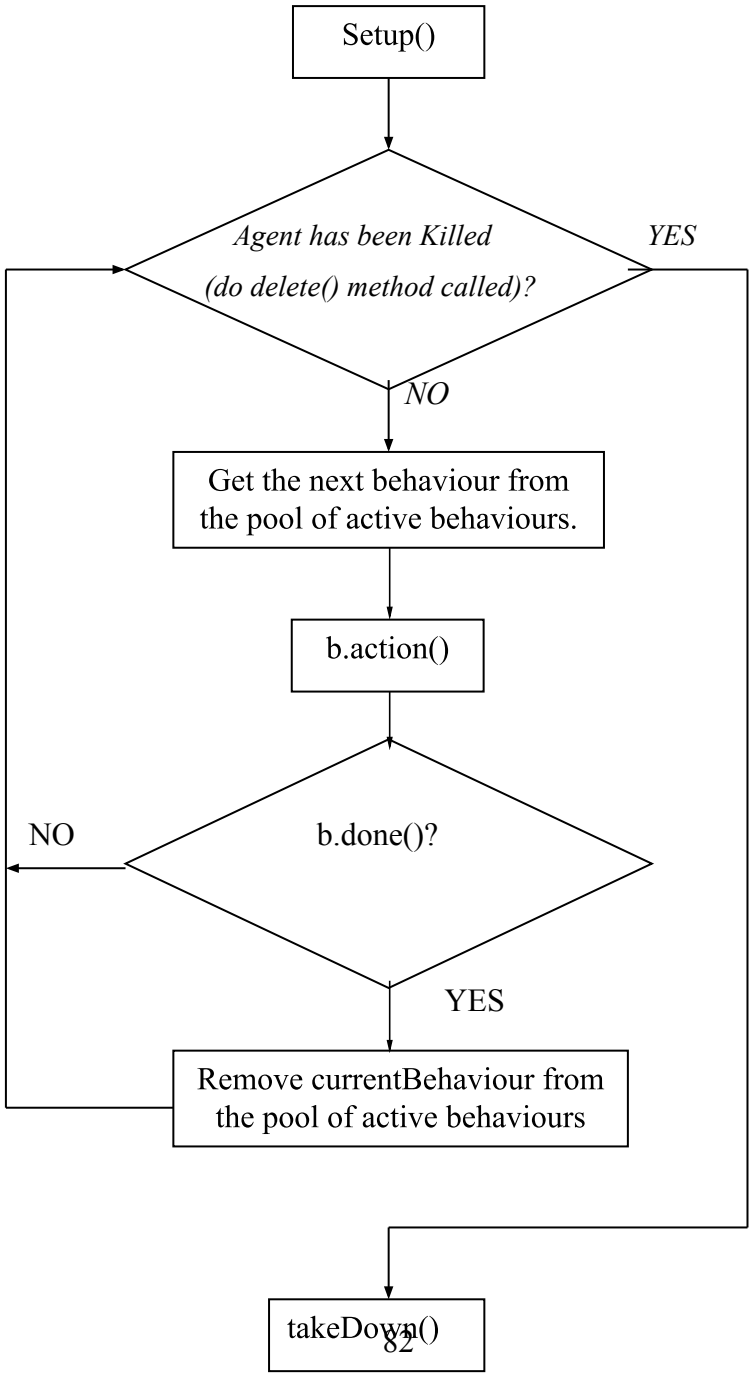
Setup()
Agent has been Killed
(do delete() method called)?
YES
NO
Get the next behaviour from
the pool of active behaviours.
b.action()
b.done()?
NO
YES
Remove currentBehaviour from
the pool of active behaviours
takeDown()

Figure 4.4: Agent Thread path of execution

It is important to stress that a behaviour like that reported below prevents any other behaviour to be executed since its action() method never returns.

```
public class OverbearingBehaviour extends Behaviour {
        public void action() {
while (true) {
        // do something
        }
}
public boolean done() {
        return true;
}
        }
```

When there are no behaviours available for execution the agent's thread goes to sleep in order not to consume CPU time. It is wakening up as soon as there is again a behaviour available for execution.

4.9.2 TYPES OF BEHAVIOUR

There are three type of agent behaviour:

1. **One-shot behaviours**: One shot behaviour complete immediately and its action() method is executed only once. The jade.core.behaviours.OneShotBehaviour already

implements the done() method by returning true and can be conveniently extended to implement one-shot behaviours.

```
public class MyOneShotBehaviour extends OneShotBehaviour {
        public void action() {
// perform operation X
}
        }
```

Operation X is performed only once.

2. Cyclic behaviours: It never complete and whose action() method executes the same operations each time it is called. The jade.core.behaviours.CyclicBehaviour already implements the done() method by returning false and can be conveniently extended to implement cyclic behaviours.

```
public class MyCyclicBehaviour extends CyclicBehaviour {
        public void action() {
// perform operation Y
}
        }
```

Operation Y is performed repetitively forever (until the agent carrying out the above behaviour terminates).

3. Generic behaviours: It embeds a status and executes different operations depending on that status. They complete when a given condition is met.

```
public class MyThreeStepBehaviour extends Behaviour {
        private int step = 0;
public void action() {
        switch (step) {
case 0:
// perform operation X
step++;
        break;
case 1:
// perform operation Y
        step++;
break;
        case 2:
// perform operation Z
        step++;
break;
}
        }
public boolean done() {
        return step == 3;
}
        }
```

Operations X, Y and Z are performed one after the other and then the behaviour completes. JADE provides the possibility of combining simple behaviours together to create complex behaviours.

4.10 GUI TO MONITOR THE ACTIVITIES OF AGENTS PROTOCOL

JADE develop some tools to support the difficult task of debugging multi-agent applications. The tools are packaged as an agent itself. It obeys the rules and communication capabilities of generic application agents.

4.10.1 REMOTE MONITORING AGENT (RMA)

The general management console for a JADE agent platform is called RMA. The RMA acquires the information about the platform and executes the GUI commands to modify the status of the platform. JADE Remote Monitoring Agent allows controlling the life cycle of the agent platform and all the registered agents. The distributed architecture of JADE also allows remote controlling, where the GUI is used to control the execution of agents and their life cycle from a remote host. RMA is a Java object, instance of the class *jade.tools.rma.rma* More than one RMA can be started on the same platform as long as every instance has a different local name, but onl/lkOy one RMA can be executed on the same agent container.

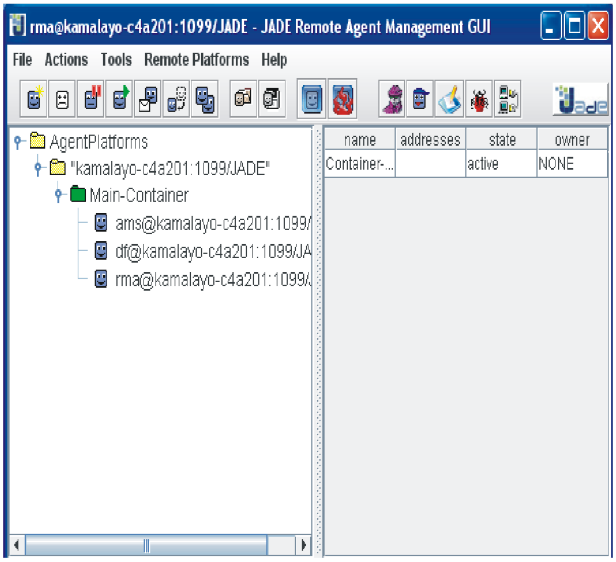

Figure 4.5: Snapshot of the RMA GUI

The followings are the commands that can be executed from the menu bar of the RMA Graphical User Interface (GUI);

1. **File menu:** This menu contains the general commands to the RMA.
   i. ***Close RMA Agent:*** It terminates the RMA agent by invoking its doDelete() method. The closure of the RMA window has the same effect as invoking this command.
   ii. ***Exit this Container:*** It terminates the agent container where the RMA is living in, by killing the RMA and all the other agents living on that container. If the container is the Agent Platform Main-Container, then the whole platform is shut down.
   iii. ***Shut down Agent Platform:*** It terminates all the connected containers and all the living agents.

2. **Actions menu:**

Action menu contains items to begin all the various administrative actions needed on the platform as a whole or on a set of agents or agent containers. The requested action is performed by using the current selection of the agent tree as the target; most of these actions are also associated to and can be executed from toolbar buttons.

   i. ***Start New Agent:*** This is used to create a new agent. The user is prompted for the name of the new agent and the name of the Java class the new agent is an instance of. Moreover, if an agent container is currently selected, the agent is created and started on that container; otherwise, the user can write the name of the container he wants the agent to start on.
   ii. ***Kill Selected Items:*** This action kills all the agents and agent containers currently selected. Killing an agent is equivalent to calling its doDelete() method, whereas killing an agent container kills all the agents living on the container and then

de-registers that container from the platform. If the Agent Platform Main-Container is currently selected, then the whole platform is shut down.

iii. ***Suspend Selected Agents:*** This action is used to suspend the selected agents and is equivalent to calling the doSuspend() method. When suspend a system agent, particularly the AMS, deadlocks the entire platform.

iv. **Resume Selected Agents:** This action puts the selected agents back into the *AP_ACTIVE* state, provided they were suspended, and works just the same as calling their doActivate() method.

v. ***Send Custom Message to Selected Agents:*** This action allows us to send an ACL message to an agent. When the user selects this menu item, a special dialog is displayed in which an ACL message can be composed and sent, as shown in the figure.

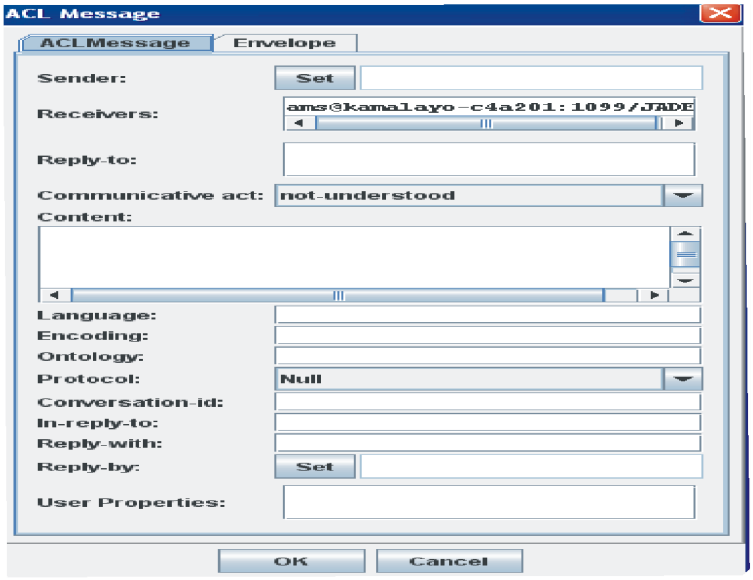

Figure 4.6: Compose and send ACL Message

vi. **Migrate Agent:** This action is used to migrate an agent. When the user selects this menu item, a special dialog is displayed in which the user must specify the

container of the platform where the selected agent must migrate. Not all the agents can migrate because of lack of serialization support in their implementation. In this case the user can press the cancel button of this dialog.

vii. **Clone Agent:** This action used to clone a selected agent. When the user selects this menu item a dialog is displayed in which the user must write the new name of the agent and the container where the new agent will start.

3. Tools Menu

This menu contains the commands to start all the tools provided by JADE to application programmers. These tools will help developing and testing JADE based agent systems.

4. RemotePlatforms Menu

This menu allows controlling some remote platforms that comply with the FIPA specifications. Notice that these remote platforms can even be non-JADE platforms.

i. ***Add Remote Platform via AMS AID:*** This action is used to get the description (called APDescription in FIPA terminology) of a remote Agent Platform via the remote AMS. The user is requested to insert the AID of the remote AMS and the remote platform is then added to the tree showed in the RMA GUI.

ii. ***Add Remote Platform via URL:*** This action allows getting the description of a remote Agent Platform via a URL. The content of the URL must be the string field APDescription, as specified by FIPA. The user is requested to insert the URL that contains the remote APDescription and the remote platform is then added to the tree showed in the RMA GUI.

iii. ***View APDescription:*** This can be used to view the AP Description of a selected platform.

iv. ***Refresh APDescription:*** This action asks the remote AMS for the APDescription and refreshes the old one.

v. ***Remove Remote Platform:*** This action permits to remove from the GUI the selected remote platform.

vi. ***Refresh Agent List:*** This action performs a search with the AMS of the Remote Platform and the full list of agents belonging to the remote platform is then displayed in the tree.

4.10.2 MEETING SCHEDULER

This is an application that uses JADE for message exchange and for the implementation of the interaction protocols. It implements a meeting scheduler agent that helps a user in scheduling meetings with other users. This sample application is based on the one that was successfully used at the FIPA Interoperability. It allows and to try check:

a. the registration of an agent with the default DF (Directory Facilitator) of the platform;

b. the registration of an agent with a remote DF belonging to another platform or, different from the default DF;

c. the search within the known DFs for a list of agents and their properties, in particular the list of Meeting Scheduler agents and the name of the user that they represent;
the usage of the FipaContractNet protocol, both the initiator and responder role.

When the two agents start, two login windows appear. Enter different user names (ignore the password), e.g. "Tizio" and "Ayour".

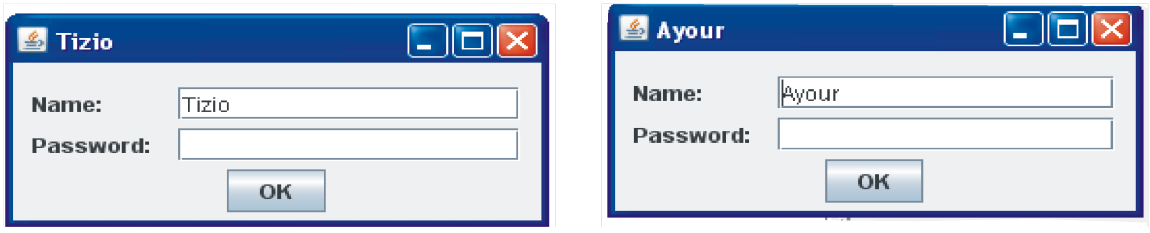

Figure 4.7 a and b: Snapshot of Meeting scheduler between Tizio and Ayour

Then, use the calendar window of Tizio to fix a couple of appointments in different dates (notice that only the day is used and not the month) with different descriptions. For instance, that Tizio has fixed two appointments for the 1st and the 2nd. Switch then to the calendar window of Ayour. From the directory menu, execute the item "Update known persons with the facilitator" such that Ayour comes to know about Tizio existence. Use the calendar window of Ayour to fix an appointment with Tizio between 1st and 4th. The ContractNet protocol should be then executed and both agents should converge to the date 3rd.

4.10.3 DUMMY AGENT

Dummy Agent is a readymade “tool” agent that allows sending/receiving custom messages, to stimulate them. The DummyAgent tool allows users to interact with JADE agents in a custom way. The GUI allows composing and sending ACL messages and maintains a list of all ACL messages sent and received. This list can be examined by the user and each message can be viewed in detail or even edited. Furthermore, the message list can be saved to disk and retrieved later. Many instances of the DummyAgent can be started as and where required. The DummyAgent can both be launched from the Tool menu of the RMA and from the command line, as follows:

Java jade.Boot theDummy:jade.tools.DummyAgent.DummyAgent

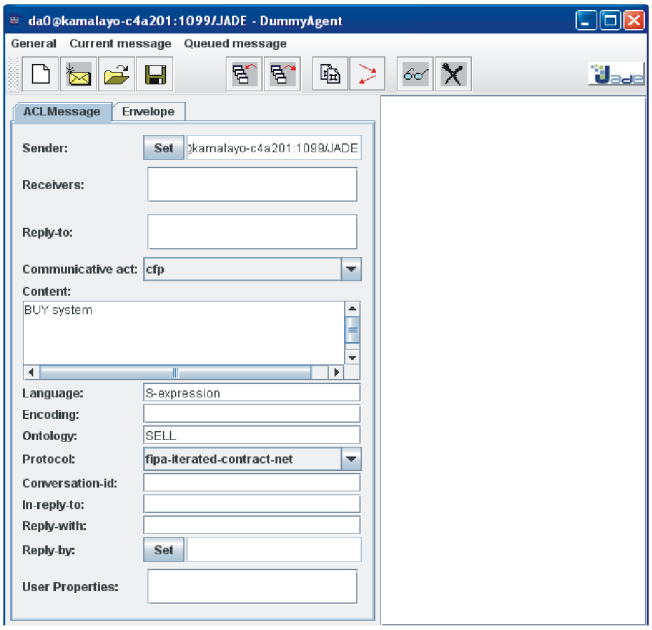

Figure 4.8: Snapshot of the DummyAgent GUI

4.10.4 GUI OF DIRECTORY FACILITATOR

The Directory Facilitator agent also has a GUI, with which it can administered, configuring its advertised agents and services. This action is actually implemented by sending an ACL message to the DF asking it to show its GUI. Therefore, the GUI can just be shown on the host where the platform (main-container) was executed. A GUI of the Directory Facilitator can be launched from the Tools menu of the RMA. By using this GUI, the user can interact with the DF: view the descriptions of the registered agents, register and deregister agents, modify the description of registered agent, and also search for agent descriptions.

The GUI also allows federating the DF with other DF's and creating a complex network of domains and sub-domains of yellow pages. Any federated DF, even if resident on a remote non-JADE agent platform, can also be controlled by the same GUI and the same basic operations (view/register/deregister/modify/search) can be executed on the remote DF.

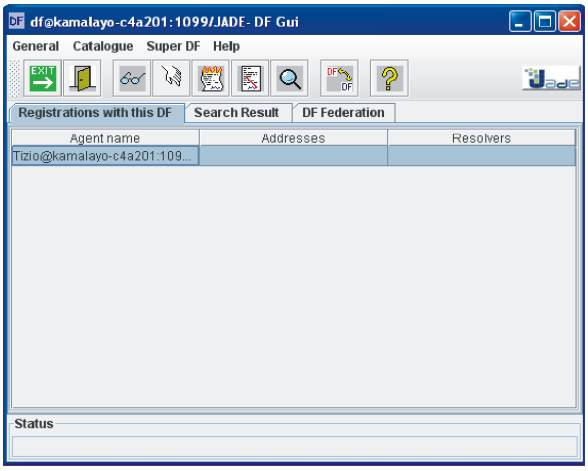

Figure 4.9: Snapshot of the GUI of the DF

4.10.5 SNIFFER AGENT

The Sniffer agents allow us to track messages exchanged in a JADE agent platform. Sniffer Agent is basically a FIPA-compliant Agent with sniffing features. When the user decides to sniff an agent or a group of agents, every message directed to/from that agent / agent group is tracked and displayed in the Sniffer Agent's gui. The user can view every message and save it to disk. The user can also save all the tracked messages and reload them from a single file for later analysis.This agent can be started both from the Tools menu of the RMA and also from the command line as follows: java jade.Boot sniffer:jade.tools.sniffer.Sniffer.

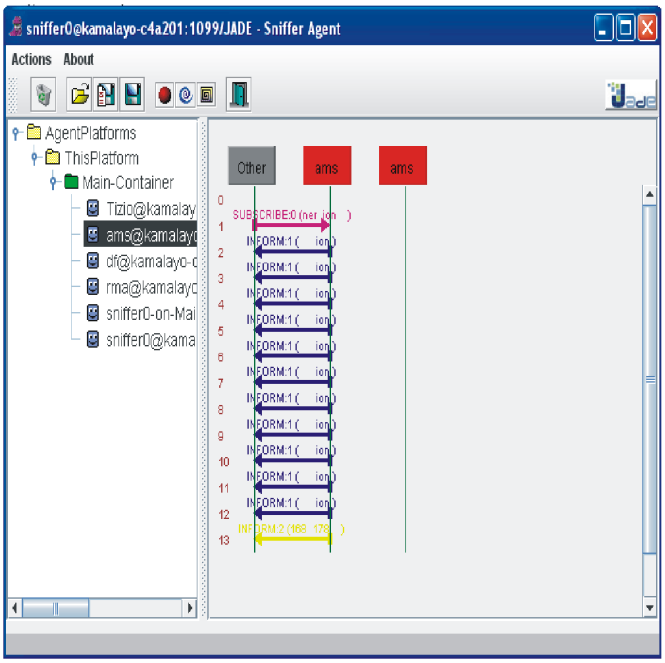

Figure 4.10: Snapshot of the sniffer agent GUI

At start up, the sniffer subscribes itself with the platform in order to be informed every time an agent is born or dies, as well as a container is created or deleted. A properties file may be used to control different sniffer properties.

4.10.6 INTROSPECTOR AGENT

This tool allows to monitoring and control the life-cycle of a running agent and its exchanged messages, both the queue of sent and received messages. It allows also to monitoring the queue of behaviours, including executing them step-by-step.

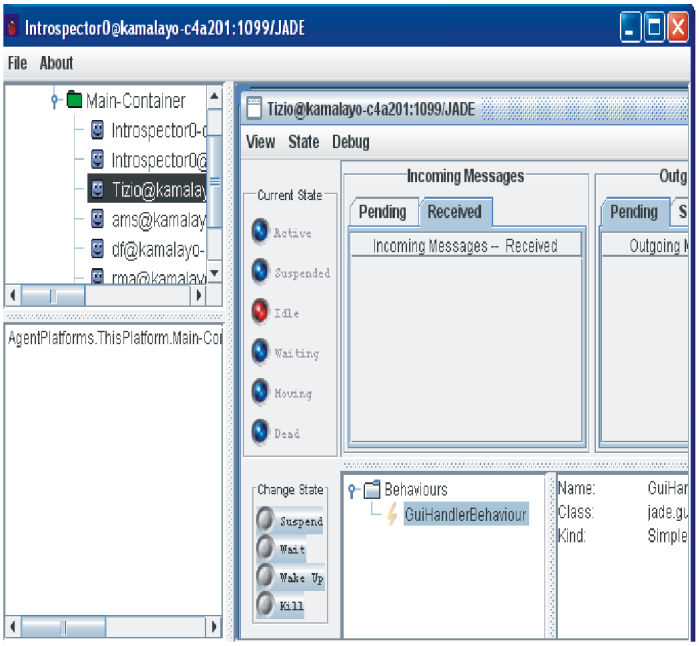

Figure 4.11 - Snapshot of the Introspector Agent GUI

Agent can be passed to the Introspector Agent in the same way as to the Sniffer Agent via the command line, or via a configuration file. The specification of performative filters, e.g. inform, agree, etc. is not supported by the Introspector Agent.

4.11 DESCRIPTION OF PROPOSED NETWORK SIMULATION

To develop and test the application, a simple network was set up. This network was designed not to be too complex so that the development was not too time consuming. The distributed Multi Agents System manages a simulated network, which is also a distributed system where each agent is tied to one node by a TCP/IP socket. Both the network simulator and the MAS are distributed systems and are able to be executed using several computers in a Local Area Network. The network simulator was implemented as independent as possible from the MAS. The application is distributed software that can be on several computer hosts with one of them acting as a front end for monitor, manage and

for inter-system communication. When the front-end computer detects any new computer, it connects it to the network. Once connected, the user can select the end-host name to communicate with. The system can be executed on different hosts, thus achieving a distributed system. The GUI of the proposed system is shown below;

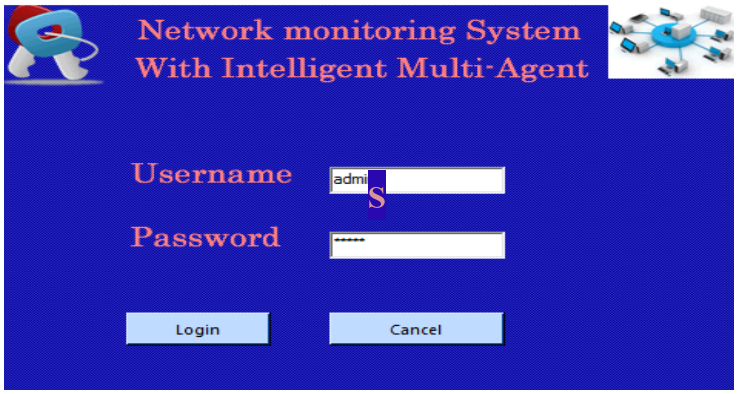

Figure 4.13: Password Confirmation Menu

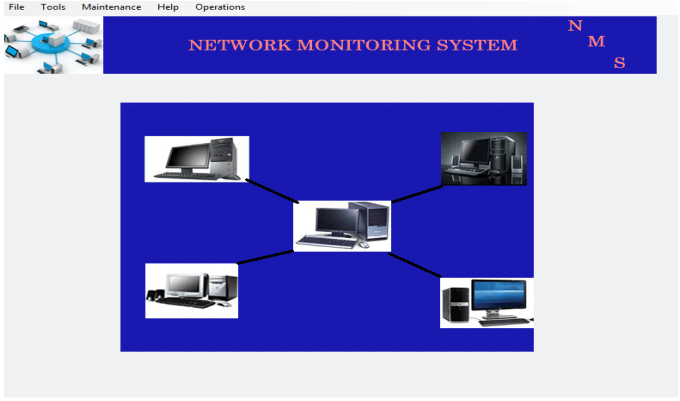

Figure 4.14: Main Menu

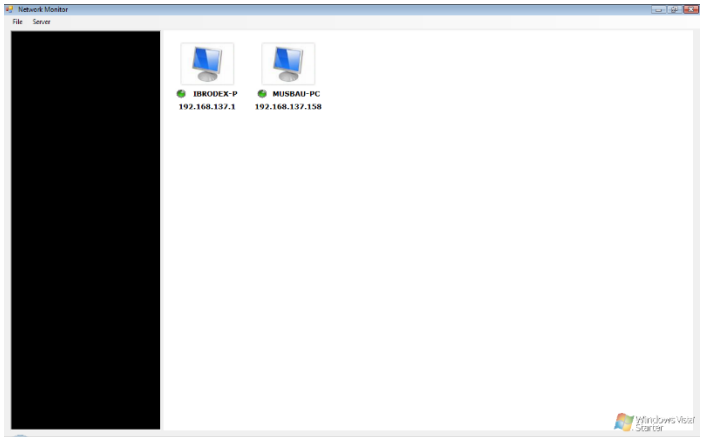

Figure 4.15: Computers Connected on Network

CHAPTER FIVE
CONCLUSION AND RECOMMENDATION

5.1 FINDINGS

Agent technology was traced from artificial Intelligence (AI) which involves how machines are made to reason and behave like human being. Software agent technology is still a niche sector of the software programming science. The major advantage of agent-oriented programming is to create a hierarchy so that the complications may be reduced. However, the following were found during the research;

a. Network management systems as represented by Simple Network Management Protocol (SNMP) are based on client-server centralized paradigm which may lead to inefficiency when the managed network are large in scale.

b. Using intelligent agents for network management is absolute feasible.

c. During the search, many platforms were found ranges from commercial to academics/research platforms.

d. Throughout the period of this research, it was discovered that JAFMAS was not available having search through its web site address several times (http://www.ececs.uc.edu/abaker/JAFMAS).

e. Though Jatlite provides essential functionality required for building a multi-agent application, it does not define methodology for specifying the social behaviour of agent.

f. Although Concordia provides a useful set of services for implementing agent mobility, security, persistence and transmission, it does not provide any methodology to specify agent in a multi-agent system coordinate, coordinate and negotiate to bring about a coherent solution.

g. Unlike others, Odyssey does not provide an extensive security mechanism. It does not also support broadcast communication and speech act.

h. MAST is implemented in C++ which makes it less potable and less multi-functional than the Java-based framework.

i. Microsoft Agents include optimal support for speech recognition in addition to the use of mouse and keyboard for input. The major limitation of Microsoft Agent is that it requires Microsoft window, Internet Explorer and ActiveX.

j. JADE can arguably be considered the most popular software agent platform available today.

k. JADE was designed with criteria more academics than industrial.

l. From a fault tolerance standpoint, JADE does not perform very well due to single point of failure represented by the Front End container and, in particular, by the MAS.

m. JADE is still an ongoing project, its development and improvement is continuing.

5.2 CONCLUSION

It was discovered that the traditional network management based on centralized architecture has some infeasibility and inefficiency problems especially in a large scale network. However, the growing interest in multi agents development platforms has led to very interesting tools. It was also discovered that using intelligent agents for network management is absolute feasible. An intelligent multi agent system technology can be used to monitor and manage possible problems in network. A research survey regarding industrial applications, multi-agent system, platforms standardization and classification were presented. Several platforms for building multi agent technologies in achieving network monitoring and management systems were reviewed and evaluated.

JADE as a software framework to facilitate the development of agent applications was later discussed. It also described its software architecture and its provided functionalities as well as its integration with some technologies especially in network management. JADE

design tries to put together abstraction and efficiency, giving programmers easy access to main FIPA standard assets while incurring into runtime costs for a feature only when that specific feature is used. JADE tries to support large Multi Agent System as possible; exploiting Jade distributed architecture, clusters as related agents can be deployed on separate agent containers in order to reduce both the number of thread per host and the network load among hosts. Finally, a distributed network prototype was designed using Visual Basic Programming language to see the possibilities of network management and monitoring system and how it can be changed to intelligent multi-agent with clear components, relationships and better control of them.

5.3 RECOMMENDATION

The system implemented is only used for network monitoring, accounting and some performance management applications. There are other numbers of potential services possible in the area of network such as fault, security and configuration management which could be improved upon later. The future work may also focuses on a platform configuration, control module and on the development of a workflow-based distributed processing engine, as well as making the agent more intelligent and giving them appropriate way of conversing with other agents. There a lot of applications in JADE that could also be implemented. However, the proposed application may also be improved upon using the same Visual basic programming or even with Java. Mobile agents and intelligent agents could also be combined in the area of proactive fault management which may allow mobile agent to have some properties of intelligence.

REFERENCES

[1] Andre Panisson, Diego Moreira da Rosa, Cristina Melchiors, Lisandro Zambenedeti Granville, Maria Janilce Bosquiroli Almeida and Liane M. R. Tarouco, (2006). *Designing the Architecture of P2P- Based Network Management systems. Proceeding of the 11th IEEE Symposium on Computers and Communications (ISCC)*.

[2] Armufo Alanis Garza, Juan Jose Serrano, Rafael Ors Carot, Karim Ramirez, Jose Mario Garcial-Valdez, Hector Arias and Jose Soria (2007). *Monitoring and Diagnostics with Intelligent Agents Using Fuzzy Logic*. Advance Online Publication.

[3] Cheikhouhou M., Conti P., Labetoulle J., and Marcus K (2007). *Intelligent Agents for Network Management: Fault Detection Experiment*. Corporate Communication department Institut eurecom: France.

[4] Christophe Andrey (1998). *Agent based network management: semester project.* Institute of Computer Science Communication and their applications. Ecole Polytechnique Federale: De Lausanne.

[5] Clyde E. Richards Jnr (2003). *Intelligent agent-based management of heterogeneous networks for the Army enterprise.* Naval postgraduate school, Monterey: California.

[6] Elegbede S. A, Rahman M. A, and Adeleke A. I. (2007). *Agent System: A Robust Technology for Intelligent Environments*. Proceeding of 1st Conference on Rule of Law and Due Process: A priority for Sustainable Development Developing Nations, Federal Polytechnic, Ede. 1-10.

[7] Faisal Alkhateeb, Zain Al-Abdeen Al-Fakhry, Eslam Al Maghayreh, Shadi Aljawarneh and Ahmad T. Al-Taani (2002). *A Multi-Agent-Based System for Securing University Campus*. *IJRRAS 2* (3), 223-248.

[8] Huawen Luo (2002). *Agent based Networks management system.* Faculty of graduate studies, University of British: Columbia.

[9] Michaud Lionel (1998). *Intelligent agents for network management: 8th semester project.* Institute of Computer Science Communication and application. Ecole Polytechnique Federale: De Lausanne.

[10] Roya Asadi, Norwati Mustapha, and Nashir Sulaiman (2009). *A Framework For Intelligent Multi Agent System Based Neural Network Classification Model.* International Journal of Computer Science and Information Security (IJCSIS) *5*(1), 68-174.

[11] Russel Stuart and Norvig Peter (1995). *Artificial Intelligence: A Modern Approach*, Prentice Hall: Ney York.

[12] Maes P (1995). *Intelligent Software: Program that can act independently will ease the burdens that computer put on people. A Journal of* Scientific Americ*an,* 273(3), 84-86.

[13] Rita C. Nienaber and Andries Bernard (2007). *A Generic Agent Framework to Support the Various Software Project Management Processes.* Interdisciplinary Journal of Information, Knowledge and Management 2(1).

[14] Ae Hee Park, So Hyun Park and Hee Youn(2007). *A Flexible and Scalable Agent Platform for Multi-agent System*. World Academy of science, Engineering and Technology, 25.

[15] Rauof Boutaba and Jin Xiao (nd). *Network State of the Art"*.

[16] Network Management: Beginners Question and Answers, available at, http://www.Geocities.com/SiliconValley/Horizon/4519/snmp2.html, Retrieved on 23/10/2010.

[17] Madejski J. (2007). Survey of the agent-based approach to intelligent manufacturing, *Journal of Achievements in Materials and manufacturing engineering, 21*.

[18] Zhong Zhang, James D. McCally, Vijah Vishwanthan and Vasnt Honavar (2003). *Multi agent System Solution for Distributed Computing, Communications, and Data Integration Needs in Power Industry*, *IEEE,* 2003.

[19] Ioannis Vlahavas et al (nd). *ExpertNet: An Intelligent Multi Agent System for WAN Management. EU INCO Compernicus Funded Project*.

[20] Wikipedia, en.wikipedia.org/wiki/software_agent: The free encyclopedia, Multi agent system, retrieved on 25/10/2010.

[21] Cougaar Software Inc.- An Intelligent Agent Technology Company, understanding the Technology series: http://www.coogaarsoftware.com Retrieved on 23/10/2010

[22] Matjin Frints (2006). *Possibilities of Peer-to-Peer Technologies in Network Management.* University of Twente: the Netherland.

[23] Insung Jung, Devinder Thapa and gi-nam Wang (2007). *Intelligent agent Based Graphic User Interface (GUI) for E-Physician.* World academy of science, Engineering and Technology, 194-197.

[24] ICN-System Management- http://www.pnk.kr/whitedell/system-1/1.html. Retrieved on 10/01/2011

[25] Cert-Coordinate center, A brief tour of the Simple Network Management Protocol (SNMP). Available at (http://www.cert.org/advisories/CA-2002-03.html), retrieved on 20/10/2010.

[26] Asante (2005). Simple Network Management Protocol, Asante Networks, Inc.

[27] Oracle Intelligent Agent User's Guide (2002), Released 9.2.0.2, Part No. A96676-02.

[28] Fabio Bellifemine, Agostino Poggi, Giovanni Rimassa (2007). *Developing Multi Agent with JADE:* Willey Series in Agent Technology, Berlin-Verlag Berlin Hrideerg.

[29] Bellifemine F., Giovanic C, Agostrio P. and Giovani R. (2008). *JADE: A Software framework for developing Multi Agent Application: Lesson Learned*. *Information and Software Technology 50*, 10 – 21.

[30] The JADE Project Home Page, available at http://www.jade.tilab.com. Retrieved on 17 March, 2011.

[31] www.cs.odu.edu/shen/intlids/assignment.html. Retrieved on 23/02/2011

[32] Intelligent Agents for Computer and Network Management, available at http//www.teachnet.edb.utexas.edu/~Lynda_abbo. Retrieved on 30/04/2011.