



HÖGSKOLAN
Dalarna

Developing a Communication Link between Agents and Cross Platform IDE

ZEESHAN AAMIR
June, 2010

Master Thesis

Nr: E3914D



DEGREE PROJECT

Computer Engineering

Program Master in Science in Computer Engineering	Reg number E3914D	Extent 15 ECTS
Name of student ZEESHAN AAMIR	Year-Month-Day 2010-06-07	
Supervisor Dr. Siril Yella	Examiner Prof. Mark Dougherty	
Company/Department Computer Science Engineering, Högskolan Dalarna, Sweden		
Title Developing a communication link Between agents and cross platform IDE		
Keywords JADE, JAVA, Runrev Revolution, Software Agent, Socket Programming, TCP/IP Layers, Server/Client, JAVA Socket, Port number, logistics.		

Note:

This thesis (project) is the part of ``BIOSIM`` project. ``BIOSIM`` is the graphical simulation software, it can simulate different types of Logistics Problems. It helps the managers or decision makers to make a decision depending on the simulation results. The main project is divided in to three parts. Part one is to separate Graphical engine form core program to improve its performance by developing a separate DLL using C++. The second part is to develop Multi Agents in *JADE* to add intelligent behaviour to graphical objects. In Third part it deals about the communication part between *JADE* and *Runrev Revolution* through TCP/IP network communication via socket. Individual students are working on this project so the contents similarity may be considered as a combine work.

ZEESHAN AAMIR

Abstract:

The main objective of this thesis work is to develop communication link between Runrev Revolution (IDE) and JADE (Multi-Agent System) through Socket programming using TCP/IP layer. These two independent platforms are connected using socket programming technique. Socket programming is considered to be newly emerging technology among these two platforms, the work done in this thesis work is considered to be a prototype.

A Graphical simulation model is developed by salixphere (Company in Hedemora) to simulate logistic problems using Runrev Revolution (IDE). The simulation software/program is called "BIOSIM". The logistic problems are complex, and conventional optimization techniques are unlikely very successful. "BIOSIM" can demonstrate the graphical representation of logistic problems depending upon the problem domains. As this simulation model is developed in revolution programming language (Transcript) which is dynamically typed and English-like language, it is quite slow compared to other high level programming languages. The object of this thesis work is to add intelligent behaviour in graphical objects and develop communication link between Runrev revolution (IDE) and JADE (Multi-Agent System) using TCP/IP layers.

The test shows the intelligent behaviour in the graphical objects and successful communication between Runrev Revolution (IDE) and JADE (Multi-Agent System).

Acknowledgments:

This thesis work is done at Högskolan Dalarna in the Department of Computer Engineering, between Feb2010 to May 2010.

First and foremost I would like to thanks to my Supervisor **Dr. Siril Yella** and Mr. Asif Ur Rahman Shaik for their support and guidelines for this project. Through their guidance and unconditional support the work and time become very smooth at Högskolan Dalarna. They created a wonderful and dynamic environment to learn in the field of Distributed Knowledge Engineering, Artificial Intelligence and Computer Networks.

I would also like to thanks Professor Mark Dougherty for reviewing this report and examining my thesis work.

A special thanks to my Director General, Mr. Anwar Amjad in IT Division at Higher Education Commission, PAKISTAN for their kind support and advice every time.

Then I would like thanks to all my friend and class fellows here in Sweden who guide me and gave me technical suggestions throughout my studies and in thesis work.

Finally, I express my greatest appreciation to my parents and my brother (Mr. Sohail Imran) for love, support and encouragement throughout my life.



Table of Contents

Note:	3
Abstract:	4
Acknowledgments:	5
Table of Contents	6
List of Figures:	8
CHAPTER 1.....	9
Introduction	9
1.1 Multi-Agents in JADE:	9
1.2 BIOSIM:.....	10
1.3 Problem Definition:.....	12
1.4 Validation:.....	13
1.5 Goals and Objectives:	13
CHAPTER 2.....	16
2.1 Runrev Revolution	16
2.2 JADE.....	19
2.3 Multi-agent systems	19
2.4 Multi Agent system using JADE.....	20
2.5 Use of Agents in logistic related problems	21
2.6 The role of humans in software systems	22
2.7 Socket:.....	22
2.7.1 Overview of JAVA Socket:.....	23
2.8 TCP/IP layer of Socket Communication	24
CHAPTER 3.....	26
Design and methodology.....	26
Pseudocode of Socket Programming to linking Runrev Revolution and JADE	26
3.1 Runrev part of execution:	26
3.2 JADE (Socket of execution):.....	27
CHAPTER 4.....	31
Establishing the connection:.....	31
4.1 Structure of Runrev Revolution:	31
4.2 Setting Socket in Runrev Revolution:	31
4.2.1 Setting Socket in JADE:.....	32
4.3 Establishing communication between BioSim and JADE:	32
4.4 Messaging System:.....	33
CHAPTER 5.....	35
Implementation:	35



5.1	Implementation of Socket on BioSim:	35
5.2	Implementation of Socket on JADE:	36
5.3	JADE multi-Agent:	36
	Architecture:	37
CHAPTER 6.....		39
	Result and Analysis:	39
6.1	Call/process the JADE directly from Runrev Revolution (BIOSIM):	45
6.2	Connectivity through TCP/IP	47
CHAPTER 7.....		51
	Conclusion and Future Works:.....	51
7.1	Conclusion:.....	51
7.2	Future Work:	52
	References:	53
	Appendix A	55
	Dictionary.....	55



List of Figures:

Figure 1.1: Connectivity Diagram between Runrev Revolution and JADE	10
Figure 1.2 : Interface view of Graphical Simulation with map.....	11
Figure 1.3: Interface view of Graphical Simulation.....	12
Figure 2.1: Code comparison with C#.Net, C++ and Java [14].....	18
Figure 2.2: Code comparison with PHP [14]	18
Figure 2.3: Connection between client and server on specific port	23
Figure 2.4: Overview of Java Sockets.....	23
Figure 2.5: OSI layers (Socket, TCP/IP).....	25
Figure 2.6 : Client and Server Socket ports	25
Figure 3.1: Java Sockets (connectivity, sends, receive and close).....	29
Figure 4.1: Stack File Structure (Runrev Revolution)	31
Figure 4.2: Proposed designed for JADE and BioSim connectivity through Socket Using TCP/IP.....	33
Figure 4.3: Proposed designed for JADE and BioSim connectivity through TCP/IP.....	33
Figure 4.4: Shows the message from revelation on JADE (vice versa).....	34
Figure 5.1: JADE Agent Behaviour (Truck, plant and field).....	37
Figure 5.2: One plant, One field, winner Trucks process for harvesting (loading, unloading and moving)	38
Figure 6.1: Runrev Revolution Design interface with (1field, 1plant, and n-number trucks).....	40
Figure 6.2: JADE bidding process (1field, 1plant, and n-number trucks)	41
Figure 6.4: Runrev Revolution (BIOSIM) simulation model (truck harvesting).....	42
Figure 6.5: JADE bidding processes for next contractor (1field, 1plant, and n-number trucks)	43
Figure 6.6: JADE Server process for socket connectivity	44
Figure 6.7: Runrev Revolution (BIOSIM) harvesting process	44
Figure 6.8: Runrev Revolution (BIOSIM)	45
Figure 6.9: JADE bidding process through shell	45
Figure 6.10: Runrev Revolution (BIOSIM) simulation model with bid winner information ..	46
Figure 6.11: Runrev Revolution (BIOSIM) harvesting process	46
Figure 6.12: JADE server on Network system.....	47
Figure 6.13: Runrev Revolution as a client on network system.....	48
Figure 6.14: Checking the delays on the network system.....	48
Figure 6.15: Checking the delays on the network system.....	49
Figure 6.16: Runrev Revolution as a client on network system.....	49

CHAPTER 1

Introduction

Graphical simulation software was developed by salixphare (Company in Hedemora, Sweden) in *Runrev Revolution* (IDE) to optimize the harvesting problem which is called “BIOSIM”.

Harvesting problem are very much complicated in logistic problems. Logistic is describe as to manage the business/military operations, such as the transportation, planning and handling the good. In harvesting problem many factors are involved, like cost, human activity, time, fuel consumption, etc. Multi Agent system are created in *JADE* for this.

An agent is a computer system that acts in an environment. It has the capability to making self-directed action in the environment in order to meet its goals. When these agents are comprise one or two are called Multi Agent System. These agents can interact with each other to find the desired goal.

The AI architecture we are implementing is based on a simplified view of human cognition known as Belief- Desire-Intention (BDI) as postulated by Bratman (1987). There are still many unexplored research areas, including the role of memory and emotion on decision-making processes, the effect of physical conditions of an agent e.g. fatigue, and the aspect of team coordination with social hierarchies where tasks need to be distributed. Some of the issues involved in linking a *JADE* based Agents system with a Graphical engine via a messaging protocol. [1]

1.1 Multi-Agents in JADE:

The Agents are created in Java Agent Development Framework (*JADE*). *JADE* is a software Development framework which makes it easy to develop the multi-agent applications in compliance with the FIPA specifications. *JADE* Agent platform try to keep the high performance with the Java language. Java library and “jar files” for Java programming. It is



used to improve the performance and make the program more efficient and intelligent. Multi Agents are developed as (Truck (contractor), Field and Plant Agents). Using these multi agents calculating the capacity of each truck, time of gathering the raw material (harvesting), plant processing time and transport raw material from the field Agent (loading raw material) to processing Plant Agent (unloading raw material). Each truck will bid for its service. Field Agent updates their progress to the control agent and the control agent will send the information to the truck agent for the availability of field. Field Agent will return its travel time and availability, travel time is 15 minutes and field will be ready to harvest in 15 minutes.

All these processes are going through Socket (TCP/IP layer) between *Runrev Revolution* (IDE) and *JADE*, which is used to make the logistic system more efficient and intelligent.

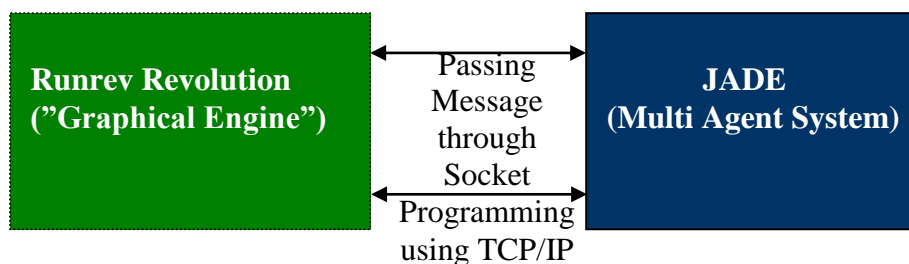


Figure 1.1 : Connectivity Diagram between Runrev Revolution and JADE

1.2 BIOSIM:

The Graphical simulation Software is Developed in *Runrev Revolution* (IDE) using scripting language as shown in Figure 1.2. "BIOSIM" simulation is build for optimise the "Harvesting Problem". BIOSIM was developed by salixphare (Company in Hedemora). It shows all important part of logistics through graphical representation which is very easy to understand. It can be used for any kind of logistic system. It shows the truck, plant, field, time, cost, harvesting tons, delivered tons, loading, unloading, distance, etc



The "logistic" comes from the French word *loger*, which means "to house", or "to lodge". Logistics is defined as the operational planning for the handling of materials, services, information and capital flows. It includes the increasingly complex information, communication and control in today's business needs.[2]

Supply chain logistics, which is concerned with the flow of goods (communication, transportation, warehousing, storage, handling, etc.). Logistic and supply management is the important part of any company's operations. The logistic discipline is considered as a complex system, the important of logistic is to fulfil the customer demands. To solve the logistic problems, that deals with the fulfilling the customer requirement at the low cost, availability and high reliability "from the customer need to customer satisfaction". The main key things have to be concerned, like delays, queue or stuck out on the right, shortest distance, etc. Right people should focus on the right process, right location, right price and right product on right time.

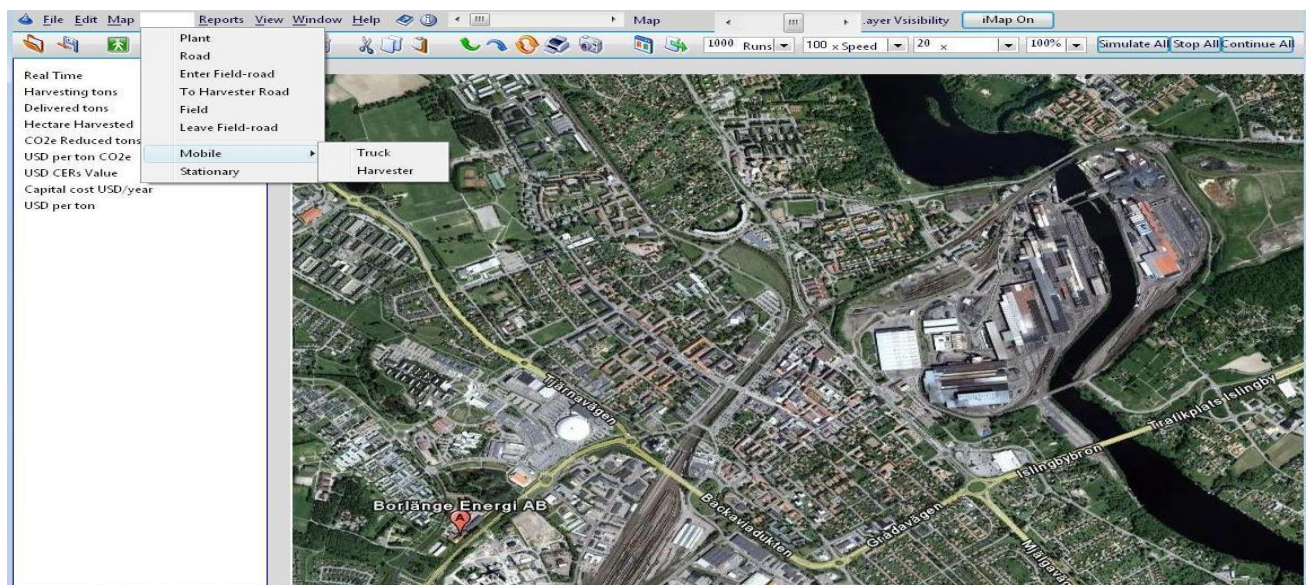


Figure 1.2 : Interface view of Graphical Simulation with map

"Biosim" simulation allows the user to design own model. In this project "1 field, 1 plant and n- number of bidders" is used to show the connectivity. User can draw roads between the field and plant according to their own choice and he/she can set the map for his own area, which shows the real time object movement for that particular area. User can see the entire task (object movement, time, cost, etc), each truck loads from the field area and unload the raw material to the plant area, after loading the raw material the capacity of field decreased.

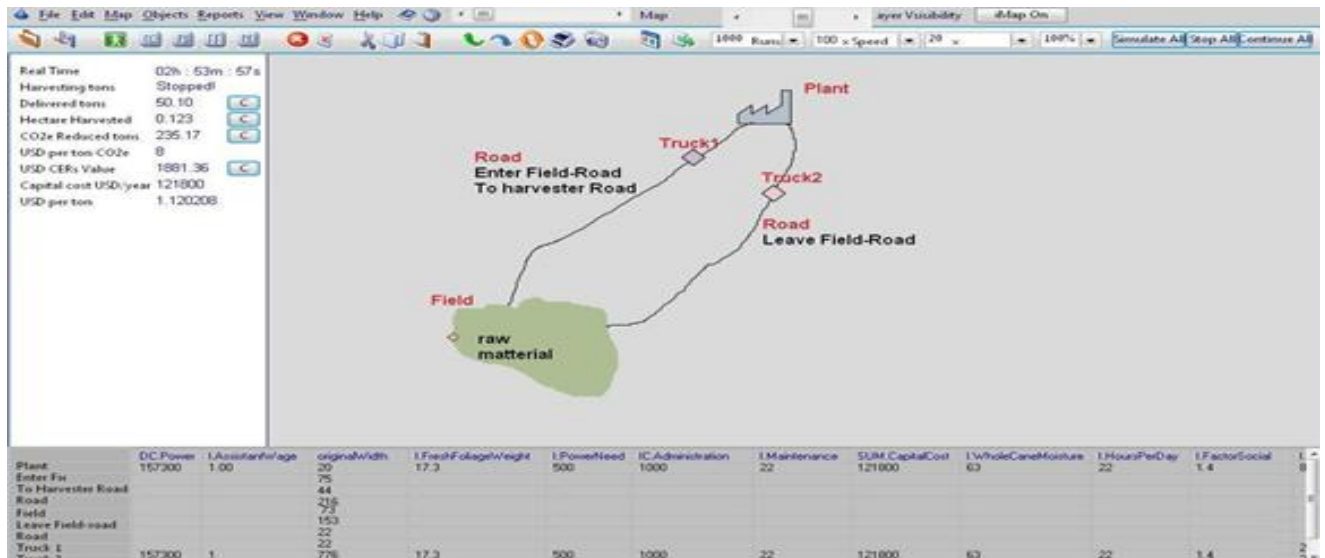


Figure 1.3 : Interface view of Graphical Simulation

1.3 Problem Definition:

Today, several industries or organisations may be small or large, have their own logistics problems. The logistics transportation is faced with constantly spreading worldwide trading and good flow. This global context required distribution and high flexibility in the transportation scheduling system, as can be provided by multi-agent system, as demonstrated by other approaches such as (Fischer,Muller,&Pischel 1995). Lot of work has been done to use multi agents in logistic problems, but this work is quite different what has been done earlier with agents. Here we are using two different platforms one is *Runrev Revolution* (IDE) and other is *JADE* (Agents). The graphics part has been developed and known as “BioSim” throughout this work.

The aim of this work is to develop a communication link between two platforms, graphical engine and *JADE*. The communication between *JADE* and *Runrev Revolution* “BioSim” had been done though Socket using TCP/IP.

To make "BioSim" simulation program more intelligent with the help of *JADE* through TCP/IP. To do this "BioSim" need some modification. The technique is used to add the socket programming. But the question is where the modification should place in "BioSim"? Should



some new button to be add with the existing system? How does the communication work? How *JADE* multi-Agent works? To find the solution of all these questions, the structure of *Runrev Revolution (IDE)*, *JADE* multi-Agent and Socket programming should be concerned. Therefore, a messaging protocol needs to be developed that will allow to exchange the information between two platforms/systems. Two type of messaged required; server message and client message.

1.4 Validation:

To validate the problem we have test the communication using one sample case of harvesting logistic problem. The logistic problems are typically complex. Logistics manage processes of harvesting, plant processing time, transporting the raw materials, coordination of information, storage, warehousing, etc without any delay or queuing.

During the validation process test case is executed, where 1 plant, 1 Field and n numbers of Trucks contractors are implemented in the scenario. Depending upon the plants bid invitation the contractors will make the bid and best will be selected and the plant will assign the work (field address and distance etc). Later on with this communication link, more cases or scenarios are tested. See section (Future Work).

1.5 Goals and Objectives:

The logistic simulation (Optimization techniques) is used to increase the efficiency and performance of the industrial operations.

Naturally this Simulation is used to link with the *JADE* multi Agents through Socket but the primary focus is decision Support, efficient and intelligent decision to full fill the user need.

JADE part deals bidding, plant agent need bidding information from truck contractor. This winner information is sent from plant through *JADE* to the *Runrev Revolution* (“*BIOSIM*”). The Field processing time is 15 minutes so no two trucks can harvest at same time it should be one after another. In case if there are two fields then truck drivers has to communicate with



each other to avoid collisions and waiting time. The “BIOSIM” is graphical interface; it improves their performance with the intelligent behaviour of *JADE*.

The user make graphical model (on map if required) and run their simulation to see the advance planning for their system. During this simulation the object works with the help of inside script.

Hare we design object (in graphical simulation) to behave intelligently and autonomously with the help of *JADE* Multi-Agent System. We are using (1Plant, 1field, n-trucks) at initial in this thesis to check communication and behaviour of the Agents on Object through TCP/IP. Both platform communicate with each other on same machine or over Networks via messaging protocol. First part deal with the bidding to optimize the truck on field and plant for harvesting.

When we increase the cases to (1 plant, n fields, n trucks(contract)) in future, it send the contractor quantity with prises and time availability for bidding and number of field and their route information to *JADE* through TCP/IP. *JADE* calculate the process and send the initial bid winner information with direction towards different field to *BioSim*. All these intelligent work will be done through *JADE*, their graphical representation will be shown on *BioSim*. Same as (n plant, n fields, n trucks(contract)) take the information from the *BioSim* and process all the information on if then rule and send back the information to *BioSim*, which shows on the GUI. When the harvesting on the field will be completed on *BioSim* side it send the information to *JADE* with the plant and truck which are busy with this field, then *JADE* process that request in their own define rule. Agent communication is important to optimize the route and their intelligent behaviour shows on *BioSim*.

The goal is to make the communication between *JADE* and *Runrev Revolution* “*BioSim*” through Socket. The problem can be solved through following steps

1. BIOSIM (Graphical User Interface)
2. JADE (Multi-Agent System)
3. Socket programming

All process like bidding, time availability, distance calculation, etc can be done through *JADE* and this information is sent to the GUI. Main task of this thesis work is to make the efficient communication with *JADE* and BIOSIM through Socket programming using TCP/IP.

This is the new idea for representing the logistic system to show the graphically interface with *JADE* intelligent decision and efficient communication. Graphical simulation (BioSim) was developed in *Runrev Revolution* (IDE) which has very good graphical features and very easy to understand and easily implement. This GUI separates from the core program in order to improve the performance, because implementing an algorithm in *Runrev Revolution* (IDE) will be slow (writing and cross platform external in C++ language to improve their efficiency). External functions are same as function handler. The program automatically calls for the external commands and functions within the script.

JADE MAS is used for the bidding process and intelligent decision, it improve the performance of the BioSim (*Runrev Revolution*). *JADE* communicate with *Runrev Revolution* through Socket TCP/IP.

In *JADE* each agent can be designed to act autonomously, in a *decentralized* manner, by computing a part of the schedule without needing knowledge, or reasoning. Based on our practical experience with transportation scheduling in medium and large-size shipping companies, we can testify to the suitability of multi-agent techniques to real-world problems.[3]

CHAPTER 2

This thesis work is derived from the limited researched material available on the internet and libraries for the *JADE*, *Runrev Revolution* and Socket.

2.1 Runrev Revolution

Revolution is a powerful graphical Integrated Development Environment; it has very simple programming language called Transcript. It is compatible to Java and Basic applications.

Why *Revolution* is used? It has true, native colour [4] It has three versions: MEDIA (beginner) Studio (core product) and Enterprise (Business). *Runrev Revolution* can be a major project that disposable tools for word processing, complex applications, prototypes are used and can be created by professionals, beginners can apply. This language is very easy to use; it can be used for Web-based applications to multiple platforms with a single click, and uses the power of multi-user database systems. It can for improvement of desktop applications, and internal productivity. It's simple, very easy maintenance and English language makes it easy to write and understand users. It is easily the right environment for development environment, drag and drop tools to create objects such as buttons and text box which is to be shown on the right side of the screen. *Runrev Revolution* connects directly with the object code. Then copy the object code and do it so no need to write new code for each item. There is no compile / link / run cycle test Runtime Environment, software-user interface builder and editor, and the same.

Following are the features of *Runrev Revolution* environment. [4]

- *Runrev revolution* uses Up to 10x more productive than traditional development systems. [5]
- Write up to 90% less code
- The application created by *revolution* is possible to run on web, desktop or server/client.
- It includes all the common features to any third generation language like JAVA, C and C++.



- To solve the problem in *Runrev Revolution* any other language can be used like java, C or C++ to improve the performance.
- Run and edit live
- Edit server/client scripts directly on the server/client.
- Achieve your goals in a fraction of the time it would take using any other development platform

- High-performance script execution
- Cross-platform delivery; It can be created on MAC and work on the Windows and Linux as well.
- External database access.
- Great support for Internet protocols
- A scripting language that contains about five times as many keywords and commands.
- Revolution adds a huge number of professional features, high end development tool that comes with additional business features, including true compilation, Linux, interacting with web sites and database connectivity.
- Revolution allows to Develop, test, and deploying work on multiple platforms.
- Revolution can exchange files between the two platforms. It usually takes only a little time to test and make sure everything looks just right.
- Revolution supports Unicode.
- Windows and Linux with native appearance and behaviours
- English-like programming language with 1601 commands and functions
- Support for encryption and Secure Sockets Layer (SSL)

The flexibility of *Runrev Revolution* allows users/developers on any platform and deploys true, native application or Web content to any desktop (windows or Linux). *Runrev Revolution* use English like sentences. In java the programmer having year of experience, can write the java code in short. Whereas *Runrev Revolution* can do it 90 % less. Next figures shows the comparison of *Runrev Revolution* with C++, C#.Net, java and PHP.



2.2 JADE

The Java Agent Development Environment (*JADE*) toolkit have three parts , library of classes assisting for agent development, a runtime environment with FIPA-specified for agent management services and a set of graphical tools for monitoring and debugging purposes. The goal of *JADE* is to make development easy, while the standard for compliance through a comprehensive set of system services and agents. *JADE* provides the following functions to agents [6].

2.3 Multi-agent systems

An agent is a computer system that acts in an environment. It has the capability to self-directed action in the environment in order to meet its goals [7]. It gets input from the environment and it should give back output to the system.

Intelligent agents come into view in Computer Science (CS) and Artificial Intelligence (AI) literature in the late 1980s as a result of work within the objected orientation and distributed AI fields (Jennings et al., 1998). Schleiffer (2005) states that “intelligent agent technology is the articulation of human decision-making behaviour in the form of a computer program” [7]

Multi-agent systems (MAS) are those systems which comprise on two or more agents. In MAS the Agents can interact with each other to find the result/goal, they share common information and may achieve the goal with other Agents. Cooperation is a general form of interaction between these agents, the cooperation including coordination of action and resolution [8]. Collaboration is about the allocation of tasks and resources between multiple agents, whether decentralized or centralized technologies. Coordination is about the way in which the actions of different agents in time and space are organized in order to achieve their goals. But when conflict occurs, the negotiating techniques are used to satisfy all parties.

MAS are often used in AI, distributed systems, software development, computer communications and other fields. MAS can communicate with each other to achieve a specific Objective.



Agents need each other for sharing full information and problem solving, the ability to embed multiple objective functions and the fact that design can be a wise step, as additional benefits of the MAS. Three most important potentials of MAS are, organization of the company itself, which makes it easier to understand for programmers and analysts to their function and behaviour. Second, problem-solving in the system is based on problem solving in the organization (decentralized: no "agent" includes the own system). Third, as autonomous agents, and is always active, the system is responsive to changes and problems. [7]

The methods implemented to achieve the communication and coordination between the essential characteristic of MAS (Odell et al. 2002): "designing an agent-based system is not just about designing the agents; it is also about designing the agent environment and interaction." Agents exchange the message and communication with each other. Agent coordination - or "interaction" - concerns the mechanism to organize by the agents themselves, to describe efforts to bring the system problems. The agent communication is communication from both linguistic semantics and protocols dialogue dominates. Semantics refer to the importance that is in a language or code to express. A dialog protocol is a series of Rules for the dialogue between two or more communicating agents (Endriss et al., 2003).

2.4 Multi Agent system using JADE

A lot of researches and commercial organizations are involved in the development of agent applications and a large number of agent building tools have been developed [10]. *JADE* is a software framework which makes it easy to develop the multi-agent applications in compliance with the FIPA specifications.

JADE agent platform keep the high performance of a distributed agent system implemented with Java language. *JADE* uses the agent model and Java implementation which allows the good runtime effectiveness, software reprocesses, agent mobility and understanding the different agent architectures.



In these days, the agent-base technology are to be consider the most capable to implement worldwide applications that often must work across corporation and continents and inter-operate with other varied systems. It can manage complex applications.

2.5 Use of Agents in logistic related problems

Most of the logistics model has central information processing. However, the key solution is to a high complexity (Marike and McFarlane, 2005). A distributed solution, including MAS is an example of the suitable circumstances where there is a traditional centralized solution is less than an adequate and dissemination of information and decision-making process seems necessary. McFarlane and Marike identified three possible objects in a distributed solution approach. First among these is a centralized solution of (theoretical) is not practical.

At any moment, no decision nodes are available, only a portion of the requested information is utilised for decision-making. Impractical is the second feature. Although all data available for each decision is crucial for the implementation of synchronized, centralized decision-making may inhibit a centrally-based decisions. The third factor is inappropriate. Although the centralized decision making is possible and practical, it may still be inappropriate. For example, one of the benefits of distributed programming to more computing power into a better basis for decision (Monk et al., 2006). Logistics real estate problems are exactly the ideal multi-agent systems, according to Davidsson et al. (2005). Fischer et al. (1996) briefly outlines four main reasons for using MAS is congestion appealing. Firstly, the distributed nature of transportation contract jobs. Trucks and jobs is not only geographically but also has a certain autonomy area. In the second wagon handle dynamic events. Agent architectures have ability to handle this dynamic. Thirdly, the traditional methods of traffic planning, large amounts of information are managed centrally. The fourth company in the transport sector to a strong participation of negotiations and cooperation in carrying out their daily tasks.

The main specific logistics computational problem requirement has been identified as follows. When the system capabilities are set apart into independent units/agents they may be intrinsically distributed over a large network of computers. Logistic applications are especially suitable for the application of techniques such as task decomposition, where the schedule for each vehicle is computed by a single agent. Computation and system control are



distributed among the agents. Each agent can be designed to act autonomously, in a decentralized manner, by computing a part of the schedule without needing knowledge, or reasoning, about the global process of the whole system. In order to achieve a better global solution, the agents must cooperate by exchanging client orders between one other and adjusting their schedules accordingly with the goal of minimizing the overall cost. Based on our practical experience with transportation scheduling in medium and large-size shipping companies, we can testify to the suitability of multi-agent techniques to real-world problems. [3] MAS have the ability, even though the cooperative skills, not optimization algorithms.

2.6 The role of humans in software systems

An important factor for success is the role of humans in the developed system. In most Agent based systems designs, human Agents play a big role. They play no specific data in the system or monitor system status. Nissen and Sengupta (2006) that "if the data is particularly complex, novel, or risky, are the people, decision makers, supported by Smart software support systems. "They appeal to the simple operational data, which makes the People to focus more on strategic processes. People monitor operational transactions and can intervene and correct decisions when necessary. Roll program differs from the type of decision available. More strategic tasks, which have a high (ER) impact using information Collection and the preparation of these decisions, not so much by automation.

2.7 Socket:

A socket is one end-point which makes communication between two programs which are running on the same machine or over network. Sockets are used to represent the connectivity between client and server. Socket is bound to an IP address and port number so that the TCP layer can identify the application.

Normally, a server runs on a specific computer and has socket which is bound to a specific port number. The server waits from client side for listening to the socket and makes a connection request. [11] On the client end the client knows the hostname or IP address of the server and the port number of the server listening. Making a connection request the client program tries to negotiate with the server program on the hostname/IP address and port



number. When connection is establish between server and client, Client used that socket to communicate with server (read/write).

Below figure describes how the server and client negotiates with each other on TCP layer over (specific IP address) port number.

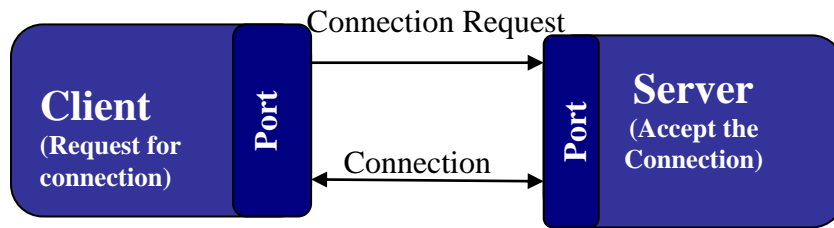


Figure 2.3 : Connection between client and server on specific port

2.7.1 Overview of JAVA Socket:

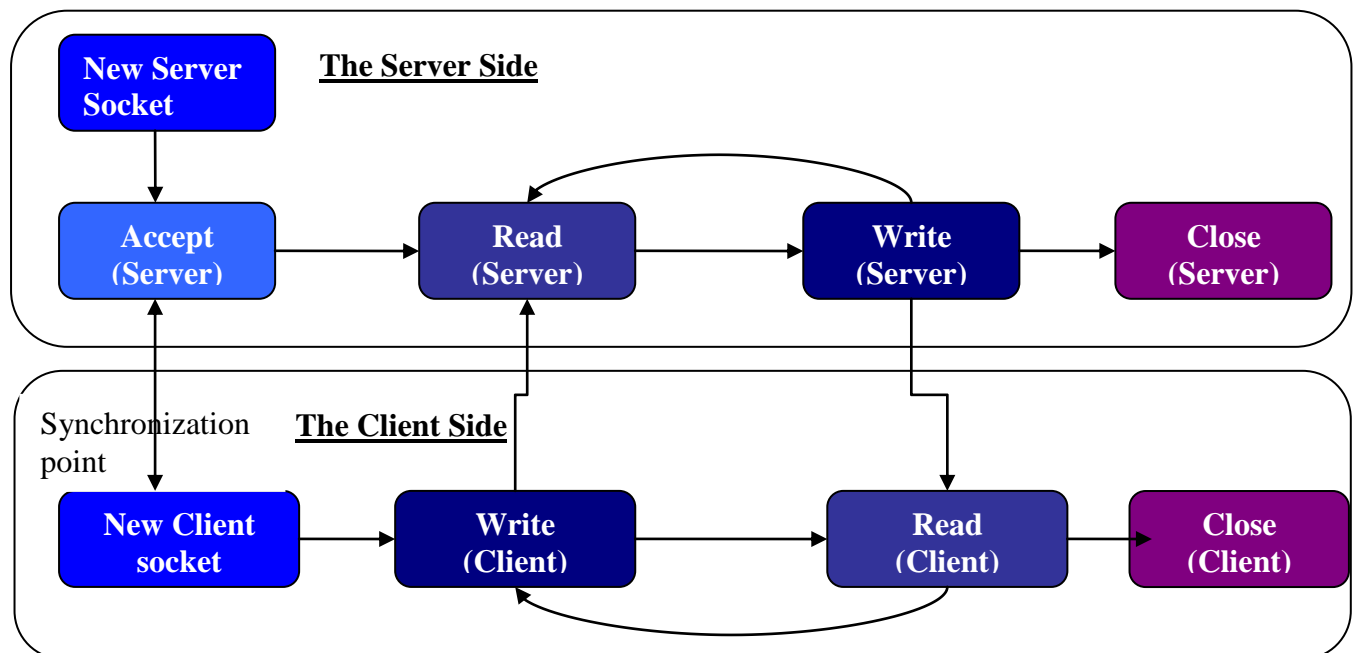


Figure 2.4 : Overview of Java Sockets

Network process in java uses sockets for two way communications running on two different machines under the same underline IP address. Socket is an end point of two way communication between two different proxies running on two different networks.

Communication link between two different machines are through port numbers.

Sockets are used to form two way communications over the networks. One of the process is involved is designate as the server and other one is client. Client actually initiates the process with the server process. Server process will bind to particular port number and wait for incoming client connections.

In java it is implemented with the socket end point for the sever process using class called server socket which is from java.net package. In the client process the class called socket also from the java.net packages.

Figure 2.4 shows that the whole process of the client and server and respective sockets related calls that different connect communicate together.

Server process we have to create server socket instants. Once server socket is created, the server process can called accept method to wait for incoming connections. (Server calling accepts as shown in Figure 2.4).

To connect with the server the client processes simply create a new socket and specify what IP address and port, server socket is blocked down to accept for. Once we create that socket in the client, server was unblocked the client process and communicate together read /write to JADE and *Runrev revolution*. Blocking accept client to makes its socket instance then we have the synchronization point on both. After that either one or both process close the socket.

2.8 TCP/IP layer of Socket Communication

Transmission Control Protocol is a connection-oriented protocol. In this protocol, the connection between the sockets must be established. All data should be read immediately, as received. One listening side called (Server) and other socket that asks to establish the connection called (client). Server socket used to accept command of the client to use the Open command. If the connection is established between socket can communicate both server and client (send and receive) with each other. Socket creates new connection with its endpoint. it use listen\read for willing to accept the connection, send\write is send or write over the



connection after the connection established and end used for release the connection. [12]

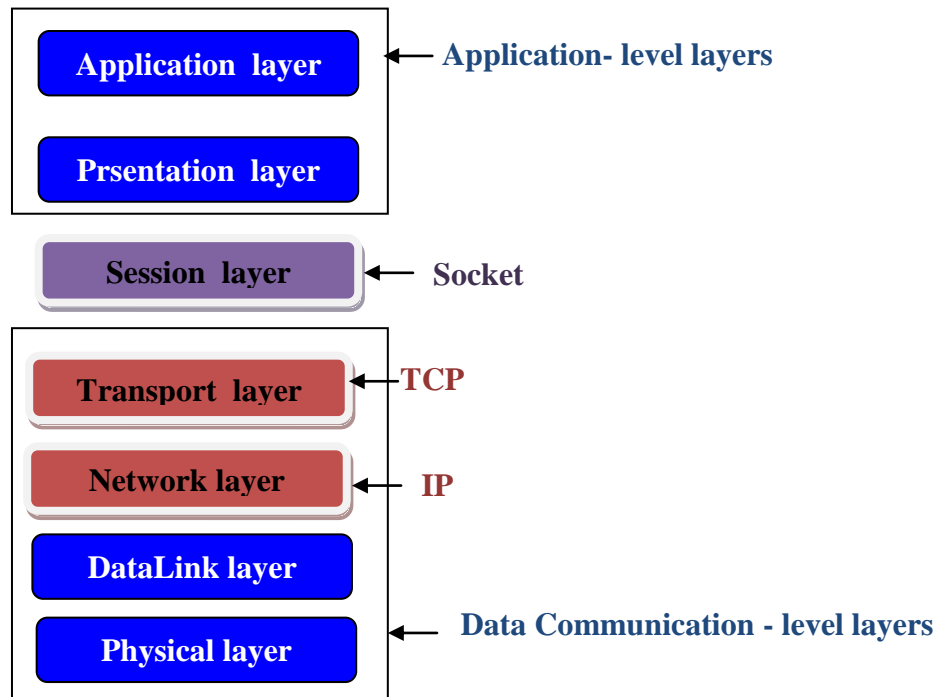


Figure 2.5 : OSI layers (Socket, TCP/IP)

Figure 2.5 shows the socket process on layers. After writing the code for Socket, which code works on presentation layer, the application layer does not know anything how socket works. Sockets reside on the Session Layer. The Session Layer is sandwiched between the application-oriented upper layers and the real-time data communication lower layers [13].TCP/IP maps the two layers. Computer transmits the data in the form of 0s and 1s.

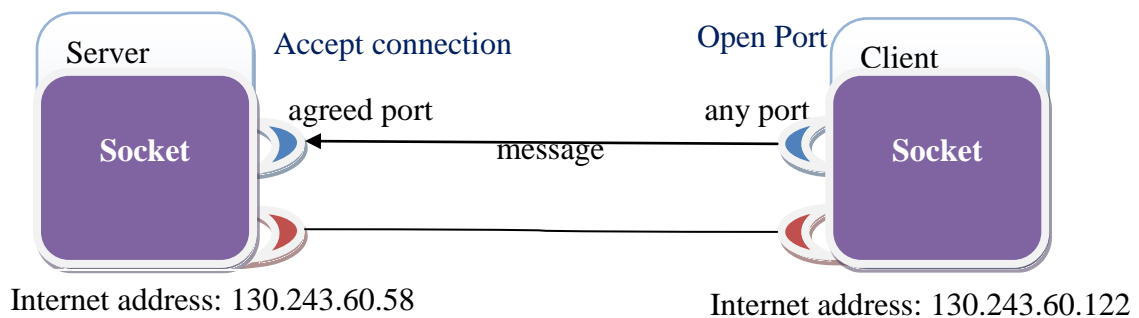


Figure 2.6 : Client and Server Socket ports



CHAPTER 3

Design and methodology

Pseudocode of Socket Programming to linking Runrev Revolution and JADE

3.1 Runrev part of execution:

Runrev Revolution uses 5 steps for Socket connectivity for server and client [13].

1. Open Socket

Open socket are used to open the connection (establish the connection) to another system on the internet or another IP network to send and receive the data. It is used on the client side.

Syntax: `open socket [to] host [: port [ID]] [with message callbackMessage]`

Example: `open socket to "192.168.0.1:8182"`

2. Accept connection

Accept command are used for accept the TCP connection from other system. The server application used the accept command to accept connection

Syntax: `accept connections on port number [with message callbackMessage]`

Example: `accept connections on port 54321 with message "ConnectToSocket"`

3. Write to Socket

Write socket used to send the data to another system via socket.

Syntax: `write value to socket socket [with message callbackMessage]`

Example: `write field "Outputs" to socket "192.168.0.1:8182"`

4. Read from Socket

Read the socket is used for get the data from another application on same system or another system using TCP Socket.

Syntax: `read from socket socketID [until string | for amount[chunkType]][with message callbackMessage]`

Example: `read from socket "192.168.0.1:8182" until return`

5. Close Socket

Close socket are used to release the connection after completing all receiving a sending data.

Syntax: `close socket socketID`

Example: `close socket "192.168.0.1"`



3.2 JADE (Socket of execution):

JADE use these command (open-accept-read-write-close) for server\client socket. First it calls open process to obtain permission for the file on client side, on server side it use accept command for establishing the connection. Once the process opened the user makes one or more class to read the data and write the data to socket (receive and send to socket). When all the process (read-write) complete, the program will finish using command close process for closing the socket.

Hare we use TCP protocol for Socket communication. Each socket has an address and port number (IP address Port number, 130.11.10.1 54321, localhost 54321). The port number between (0 – 1023) are reserved and FTP, http, etc. so the port number must be greater than 1023. Java.net package provide two classes one is *socket* which implement the client side of connection and other one is *serversocket* which implements the server side of connection

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.io.DataInputStream;
import java.io.DataOutputStream;
```

There are operations perform by socket. (Open, accept connection, send data, receive data, close connection)

1. Open Socket:

On client side create an object of Socket class. machine name/IP address and Port Number on which the server want to connect. The post must be greater than 1023. The port numbers between 0 and 1,023 are reserved for privileged users (that is, super user or root).

```
Socket Client;
Client = new Socket ("Machine name or IP address", Port number);
Public Socket (String host, int port)
throws UnknownHostException, IOException
```

On server Side for opening a socket.



```

ServerSocket Server;
try {
Server = new ServerSocket (PortNumber);
}
Catch (IOException e)
{ System.out.println(e); }

```

On server side create a socket object for accepting the connection from client for serversocket.

```

Socket clientSocket = null;
try {
ServiceSocket = Server.accept();
}
Catch (IOException e)
{ System.out.println(e); }

```

2. Create Input Stream:

On client side "DataInputStream" class are use for receive the response for the server side. This class "DataInputStream" allows to read the lines of texts. It has many methods read, readChar, readInt, readDouble, and readLine.

```

DataInputStream input;
try { input = new DataInputStream(clientSocket.getInputStream());
// bufferReader = new BufferedReader (new InputStreamReader (clientSocket.getInputStream()));
}
Catch (IOException e) { system.out.println(e); }

```

3. Create Output Stream:

On client side output stream used for send the information to the server socket, using printStream or data output Stream.

```

printStream output;
try{ output = new printStream (client.getOutputStream()); }
Catch (IOException e) { system.out.println(e); }

DataOutputStream output;
try{ output = new DataOutputStream (client.getOutputStream()); }
Catch (IOException e) { system.out.println(e); }

```



On the server side, using class `PrintStream` or `DataOutputStream` to send information to the client.

```

    printStream output;
    try{ output = new printStream (serviceSocket.getOutputStream()); }
    Catch (IOException e) { system.out.println(e); }

    DataOutputStream output;
    try{ output = new DataOutputStream (serviceSocket.getOutputStream());
    // bufferReader = new BufferedReader (new OutputStreamWriter (serverSocket.getInputStream()));
    } Catch (IOException e) { system.out.println(e); }
    
```

One thread read from the socket where as the other thread write to socket at the same time.

4. Close Socket:

On Server side (close Input and Output Stream before closing the socket).

```
Server.close ();
```

On Client side (close Input and Output Stream before closing the socket).

```
client.close ();
```

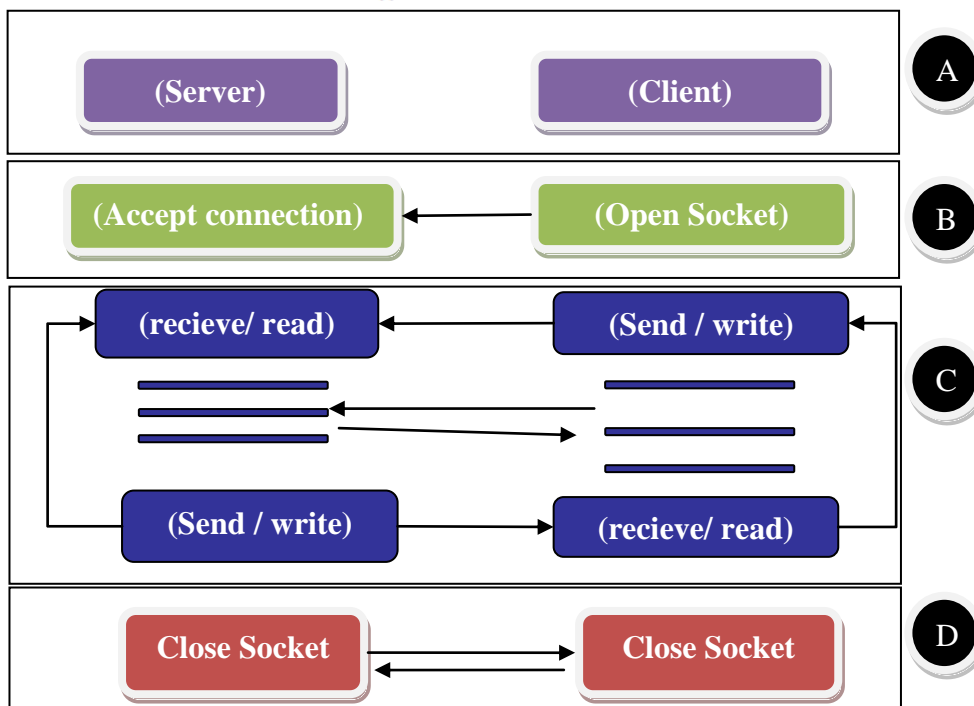


Figure 3.1 0: Java Sockets (connectivity, sends, receive and close)



In this figure **A** shows the Server and Client, **B** Shows Server listening side, the other side is on client (open socket) which asks to establish the connection, Server (accept connection) accept the request from client on same port as define on client and server. Server socket used to accept command of the client to use the Open command.

C After establishing the connection, both server and client (send and receive) with each other, until their requirement completed. **D** At the final stage when all read and write finished the connection and socket closed.

CHAPTER 4

Establishing the connection:

4.1 Structure of Runrev Revolution:

First step to create an application in *Runrev Revolution* is creating a window which is called stack. Each window in the *Runrev Revolution* is a stack. All controls are created by dragging them on to a card. Each *Runrev Revolution* file contains one or more stack, if there is more than one stack then they are categorized as main stack and sub-stacks. For easy distribution multi stack can be in single file to optimize the stack. First stack is called main stack where as other stack created in the same stack file are called sub stacks of the main stack.

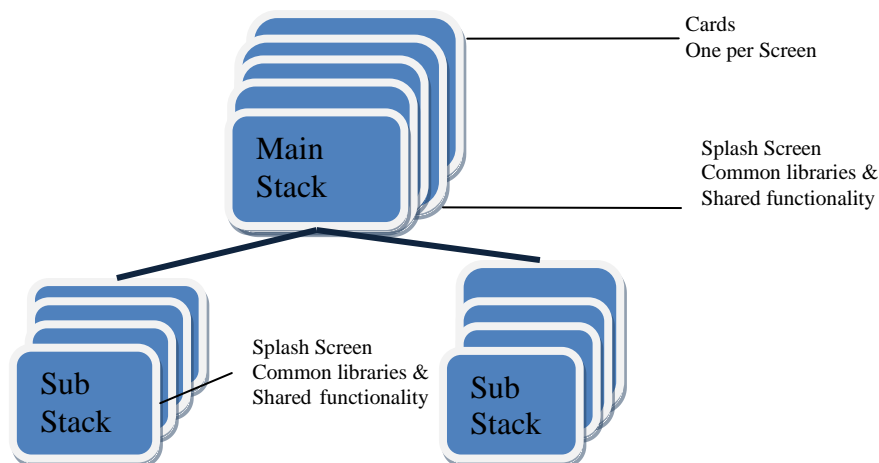


Figure 4.1 : Stack File Structure (Runrev Revolution)

4.2 Setting Socket in Runrev Revolution:

Socket coding is written under the button called (Socket).The *Runrev Revolution* “BioSim” works as a client so it use open, read, write and close socket commands to establish the communication with *JADE*. All socket related command which is used for communication is describe in Chapter 3. If-else statement is also used for object movement. Like when truckcontractor1 wins the bid then it work like that



If truckcontractor n = string from JADE

Then

Move truckcontractor n to the plant n

After reaching to the plant it go towards the field for harvesting until their entire job done.

4.2.1 Setting Socket in JADE:

Socket java class can be written in JADE, which works as a server and connected with *Runrev Revolution* “BioSim” and *JADE* Multi-Agents. The port used for socket connectivity between server and client must be greater than 1023 because the port number between 0-to-1023 are reserved for super user or root. This creates a socket object for accepting the connection from client for *serversocket*. Create Input and Output Stream for read and write the data from and to *Runrev Revolution* “BioSim”. After completing all the tasks the Input, output Stream and server socket will be closed. If then rules are used for process the command after receiving the message for BioSim through Socket. All the command related to socket in *JADE* are described in chapter 3.

4.3 Establishing communication between BioSim and JADE:

After creating Server and client, the environment is ready to communicate between BioSim to *JADE* through socket. *JADE* read the string from the BioSim and do the processes against that command using if then rule and after completing the process it sends back the result to BioSim through Socket. BioSim read that string and do the process against that string and after completing that task it sends another string to *JADE*.

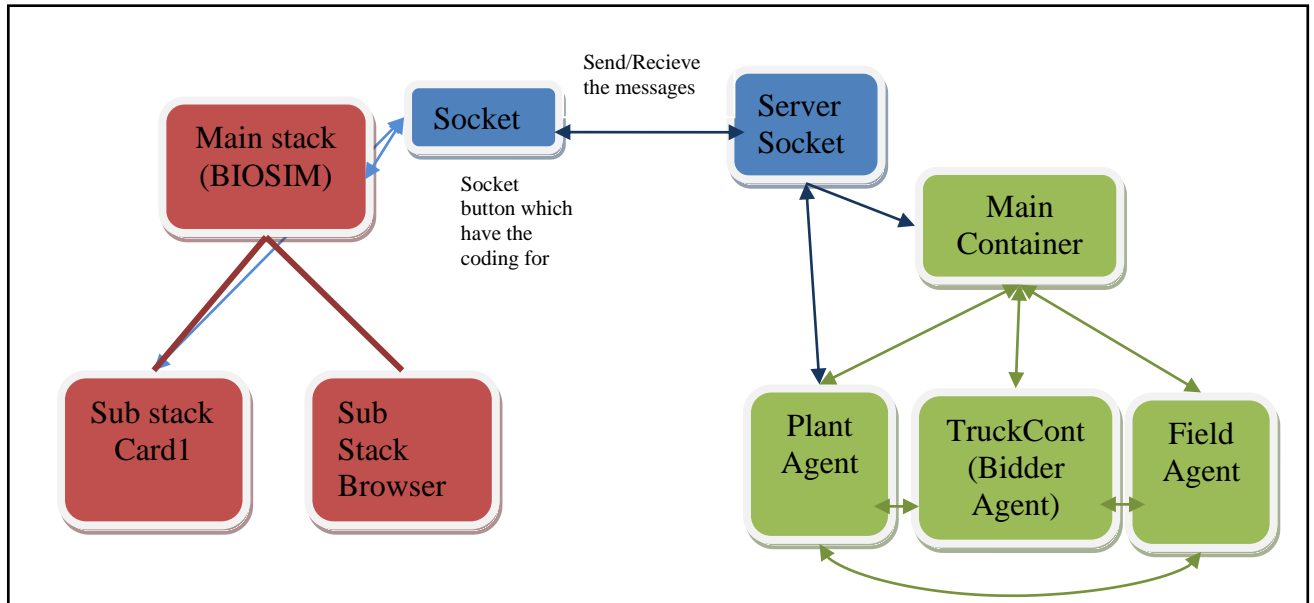


Figure 4.2: Proposed designed for JADE and BioSim connectivity through Socket Using TCP/IP

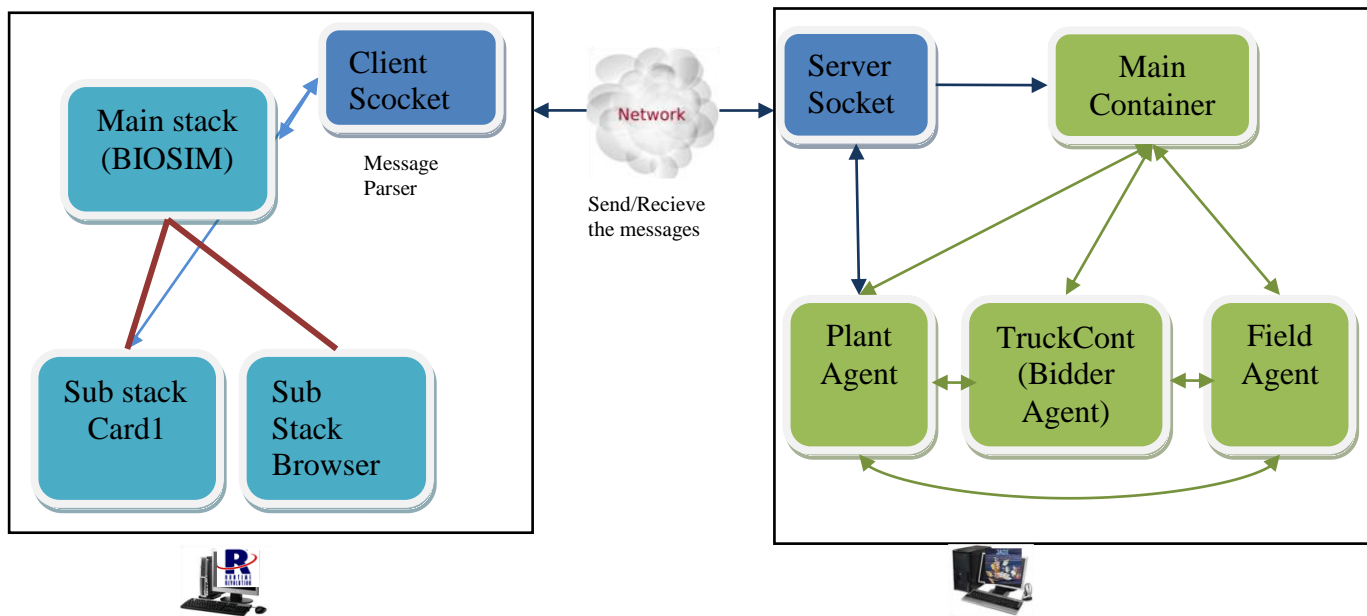


Figure 4.3: Proposed designed for JADE and BioSim connectivity through TCP/IP

4.4 Messaging System:

This system specifies messages into two types; server Messages and client messages. Messages within the server are among the main controller and the Plant, Truck Contractor and Field Agents and with Client Socket. Messages within the client deals with Main Stack to Sub



Stacks and also from the client socket with Server Socket. After a connection is established, both server and the client handshakes with each other.

Messages from the client take the form of command “strings” that instruct to the agent to perform task such as plant requirement, bidding, winner information, etc. The agent from the JADE will send the message using TCP/IP to BioSim, BioSim do the process on that such as “move to node”, etc.

```

22 {
23
24
25 //put all the agents in main container
26 ContainerController mainContainer = jade.core.Runtime.instance().createMainContainer(new Profile)
27 //Assigns the parameter for the agent plant
28 String [] param = (String.valueOf("harvest"));
29 //Initializes the plant agent
30 AgentController controllAgentController = mainContainer.createNewAgent("plant", "case1.plant", par
    controllAgentController.start();
  
```

Output window content:

```

Plant need trucks to collect items from field
plant : I need trucks to harvest 100 tons
Target is harvest
From JADE to Rev : Contractor who won in the bidding:: truckContractor1
Plant requirements completed(100 tons)

Plant need trucks to collect items from field
plant : I need trucks to harvest 100 tons
Target is harvest
From JADE to Rev : Contractor who won in the bidding:: truckContractor1
Plant requirements completed(100 tons)
  
```

Figure 4.4 : Shows the message from revelation on JADE (vice versa)

Figure shows the messages from the BioSim (*Runrev revolution*) on the JADE server and what kind of messages it sends to the BioSim (*Runrev Revolution*).

CHAPTER 5

Implementation:

This part deals with the case1 (1plant, 1 field and n number of bidders) which are written in *JADE*, it communicate with BioSim (*Runrev Revolution*) through TCP/IP.

5.1 Implementation of Socket on BioSim:

A simulation model (BioSim) is built in *Runrev Revolution*. In this the Socket Client program is written in “BioSim” which communicate with *JADE* server for sending/receiving the messages. When “BioSim” receive the string/message from the *JADE*, it do the define process/task against that string/message. One button is created in the BioSim and the coding is behind that button. When this button is clicked then it established the communication with *JADE* and do the processes, when the process completed it automatically call to the next message

Algorithm:

- Step 1: Find the IP/local host and port Number of JADE Server
- Step 2: Create a TCP Socket
- Step 3: Connect to the Socket JADE Server (Server must be up and listening)
- Step 4: Send the String to JADE Server through Socket
- Step 5: Receive the String from JADE Server via Socket
- Step 6: Do define process on after receiving the data
- Step 7: Repeat the step 4 until all jobs done
- Step 8: close the connection



5.2 Implementation of Socket on JADE:

In this the Server socket program is written in *JADE* side which accept the connection from BioSim (*Runrev Revolution*) for read/write the data. *JADE* Server must be up and listening for request from the client or server process can be call from “BIOSIM” directly. After establishing the connection, “BioSim” (*Runrev Revolution*) write to *JADE*, and *JADE* do the process on that request and send resulted request back to “BioSim”.

Algorithm:

- Step 1: Find the IP/localhost and port Number of Server
- Step 2: check the time out for connection
- Step 3: Create a TCP Server Socket
- Step 4: Bind to server Socket to server IP and Port (the port on which client will connect)
- Step 5: Accept the connection from “BioSim”
- Step 6: Send/receive the string to “BioSim”
- Step 7: Do the define process against the received string and return the result string
- Step 8: Do the step 6 until all jobs done
- Step 9: close the connection with “BioSim” Client

5.3 JADE multi-Agent:

One program is written in *JADE* which has 1Plant, 1Field and n number of Trucks Agents (Contractors). Bidding part takes place between the Truck Agent (Contractor) and Plant Agent. Plant is working as a controller agent , decides the bidding on different cases.

1. When the field is available and the plant sees the availability of truck, if so the plant agent sends the truck to the field agent for Harvesting. So in this case plant chooses the least available time.
2. Plant checks for least price, for this plant collect all the quotations from the contractor and choose the least price.
3. Plant collects all the quotations from the Truck Agent (contractor) with the prices and availability. After that plant checks the least available time and price then plant again



checks the next least availability and respective price. The plant Agent repeats the process until it finds the best price and availability.

Architecture:

Three Agents are made in *JADE*, these multi Agents (Plant, Field and Contactor (who owns the truck)) works different kind of tasks. All agents register their service with each other in directory facilitator. Plant Agent search for trading with the contractor and start bidding process. After that contractor Agent send bids randomly (with price and availability as a message) to fix the number of contractors and the bid prices. When plant Agent receives the bids, it starts the process and compares the least price and least available time. When any contractor wins the bid the plant Agent sends the message to the field and notifies the field and truck Agent.

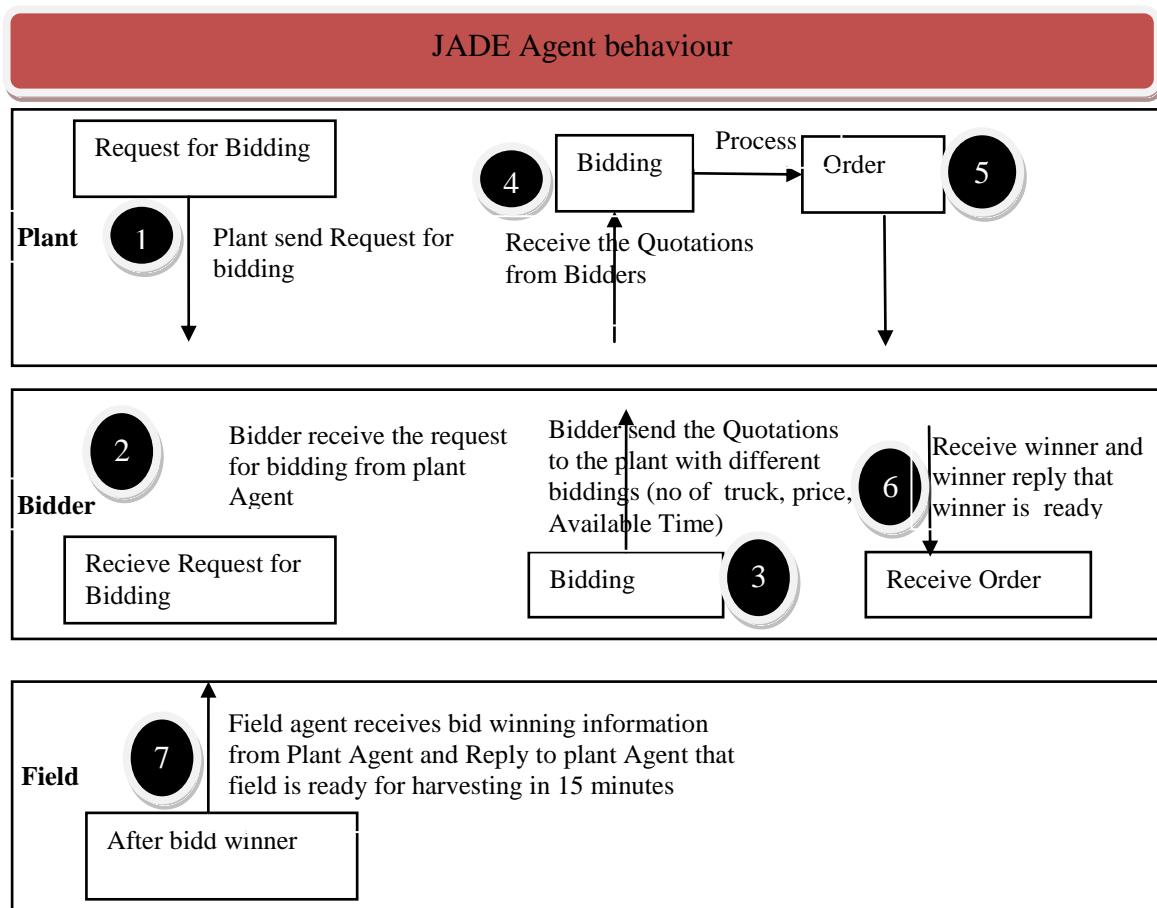


Figure 5.15: JADE Agent Behaviour (Truck, plant and field)



Case 1	
Agents	1 plant, 1 field, n Trucks
Target	100 tons Raw material collection notified by Plant
ALL Agents	Need to register their service in order to communicate with each other.
Plant Agent	Search for trading in DF. Start the bidding process. Fix the price and availability or just deal with price only. Randomly set the price for all contractors. Notify to field and plant Agent after completing bid process.
Truck Contactor Agent	Have negotiated the price. Send the quotations/bidding to plant Agent with price and availability. Send the least price and availability. Winning contractor collect the item from field. Winner truck loads the raw material for the field Agent and unloads into to the plant Agent.
Field Agent	Reply to Plant Agent about finish harvesting and ready to work.
Time	Time between plant to field is 15 min.
Capacity	All Truck have same capacity (25 tons)

Case 1: This table explains the whole process as described before.

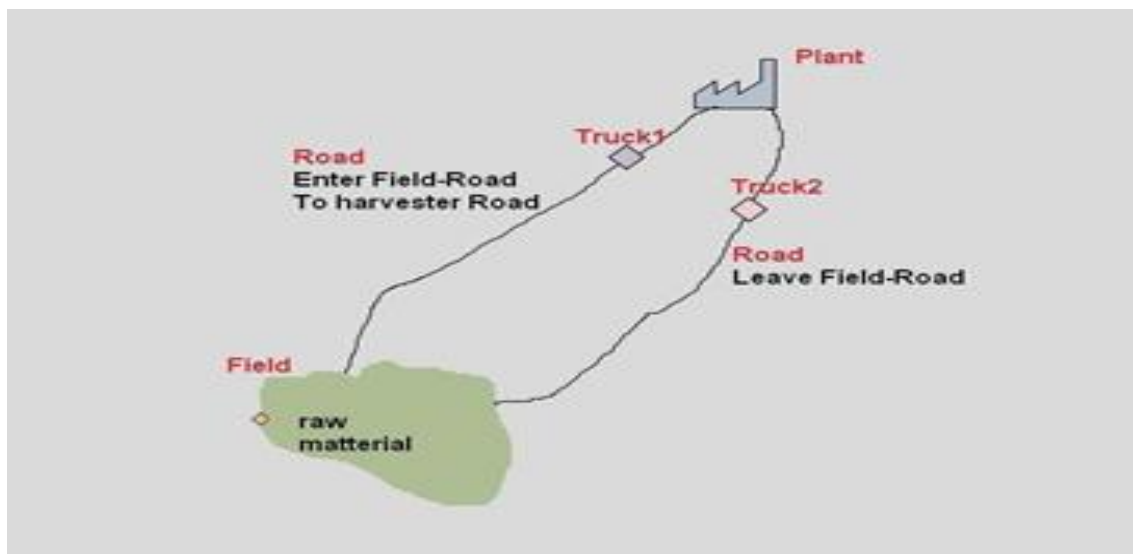


Figure 5.2 16: One plant, One field, winner Trucks process for harvesting (loading, unloading and moving)

This figure shows the behaviour of one plant, one field and the winner trucks who win the bid on BioSim. “BioSim” receive the bid winner information and call that bidder for harvesting. Only those bidders (trucks) who win the bidding do their job.

CHAPTER 6

Result and Analysis:

All the Agents (1plant, 1 field and n-number of trucks) are created, implemented and tested in *JADE* (Multi Agent System) which is the powerful tool for Agent communication. They are graphical represented in “BioSim”. BioSim is connected with *JADE* through Socket as a client/server. The goal of this thesis was achieved to make the efficient communication between *JADE* and BioSim (*Runrev Revolution*).

To solve the problem of the original designed model, graphical engine “BioSim” (1plant, 1field and n-number of trucks) was created. This “BioSim” is working as a client of TCP/IP environment. 1plant, 1field and n-number of trucks were created in the *JADE* environment and it is working on a Server. When the connectivity is established between BioSim and *JADE* server on a specific port, BioSim send the message to the *JADE*, *JADE* Server receive the command from the client and process their request using if then rule. *JADE* send back the result information to BioSim.

When the BioSim receive the message from the server it shows the representation of result on graphical engine, bid winner truck move towards the plant and field for harvesting the biomass and unload this to the plant. All the delivered tons on the plant continuously show on the side box of the graphical engine.

These all processes make the program efficient, intelligent and fast. All the process with figures and their description are shown as follows.

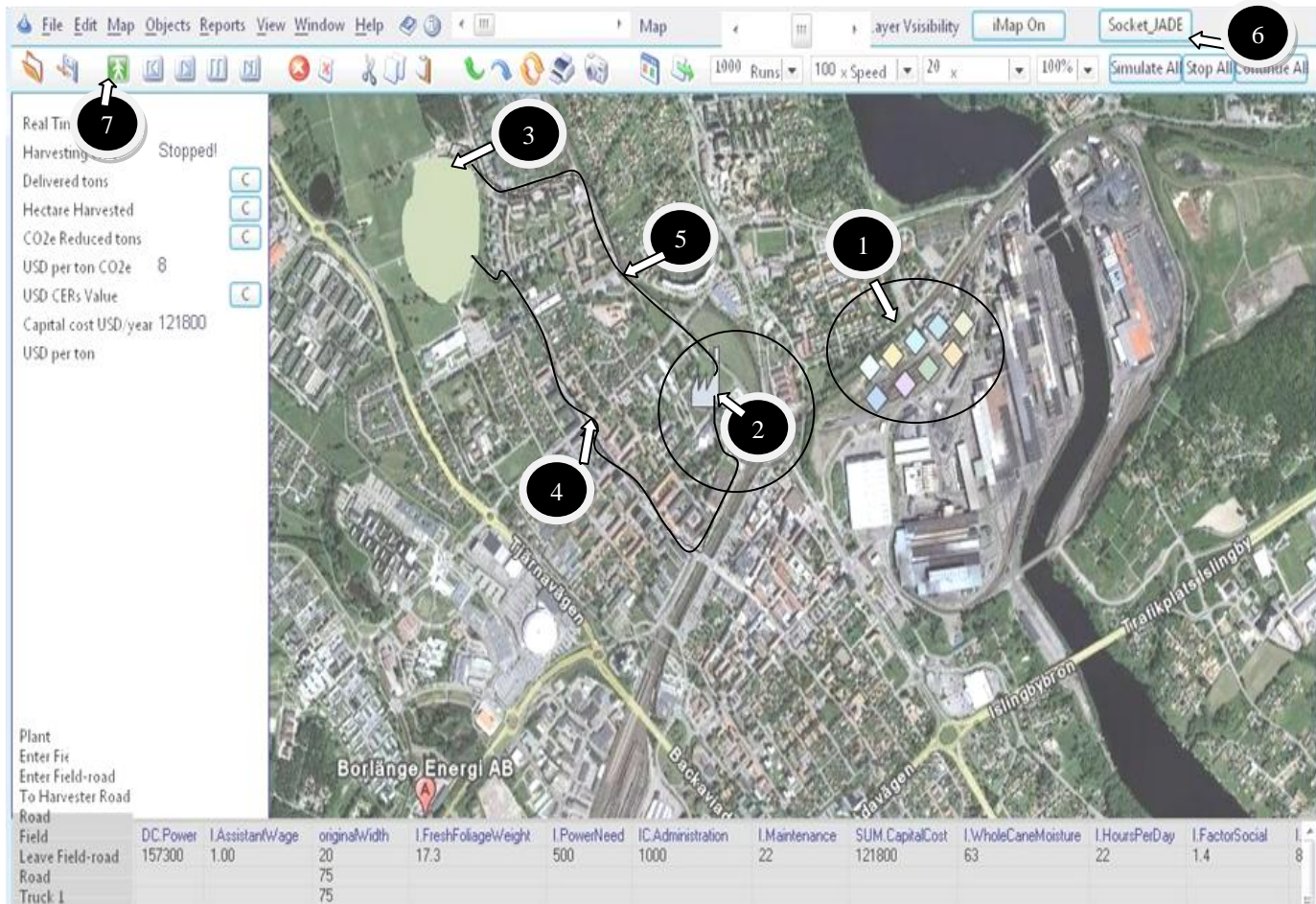


Figure 6.1 Runrev Revolution Design interface with (1field, 1plant, and n-number trucks)

Figure 6.1 shows the interface of BioSim with different objects. It consists of one field, one plant and n-number of trucks. The plant is not so far from the field. n number of trucks are taken to check the randomness.

- 1 Shows the truck waiting for the bidding, (In case more truck required it can be add dynamically)
- 2 Shows the Plant
- 3 Shows the Field
- 4 Shows the road towards the plant from field (truck leave field after loading the Bio mass towards the plant for unloading)
- 5 Shows the Road toward the field (truck enter this road loading Biomass toward the field)
- 6 Shows the button (Which makes the connection with JADE server for send/receive the information)
- 7 Shows "Run" the simulation (to start the process)

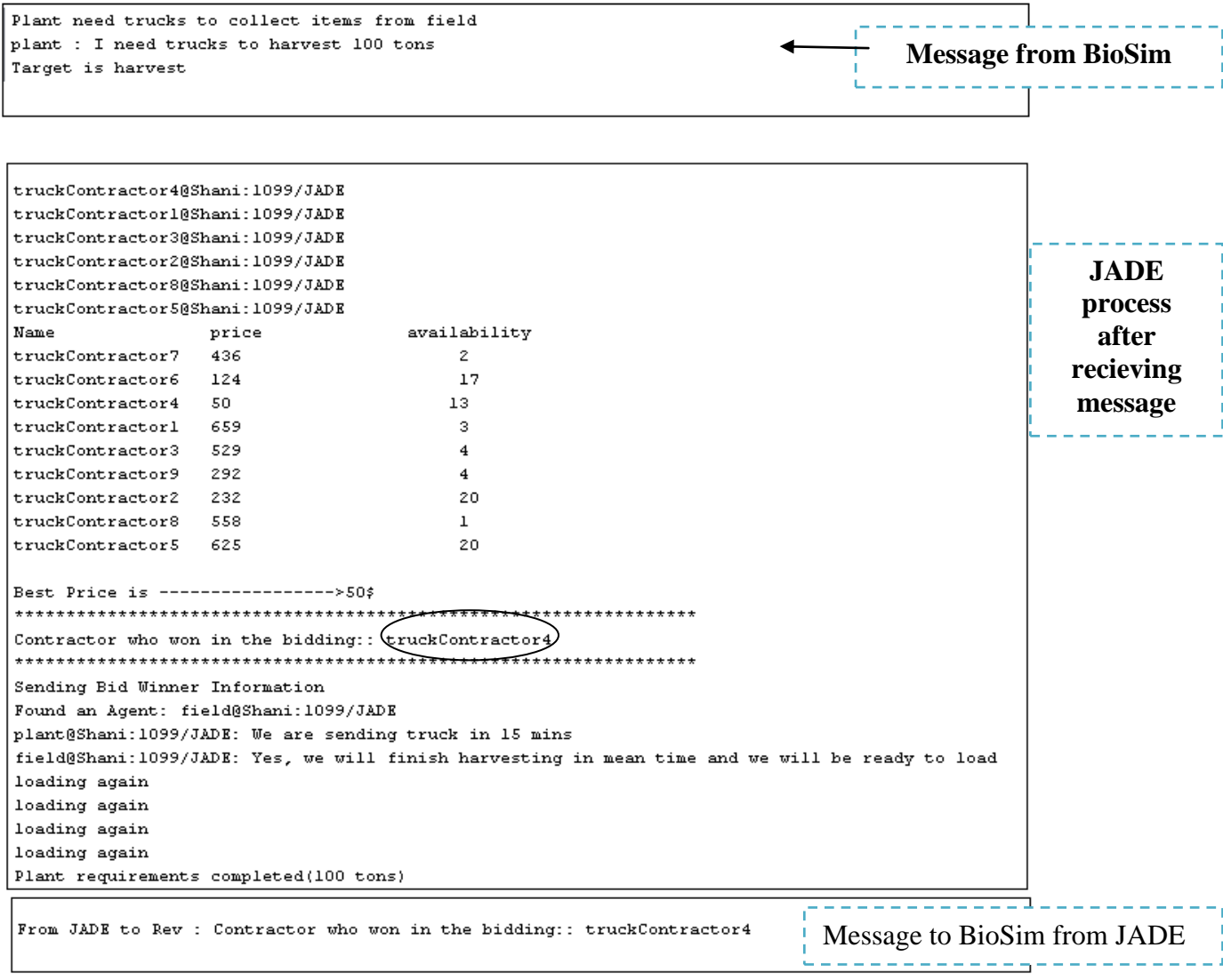


Figure 6.2 JADE bidding process (1field, 1plant, and n-number trucks)

This Figure 6.2 shows the interface of Message received from BioSim for bidding to *JADE*. It consists of one field, one plant and n trucks, the truck Agents (bidder) bid for their services. The bidding are generating randomly after that plant execute all these bidding process. In this case n number of different bidders, bids for their services, plant processes the bidding and it send the order to the least price bidder. After completing all the process *JADE* send the message to BioSim.

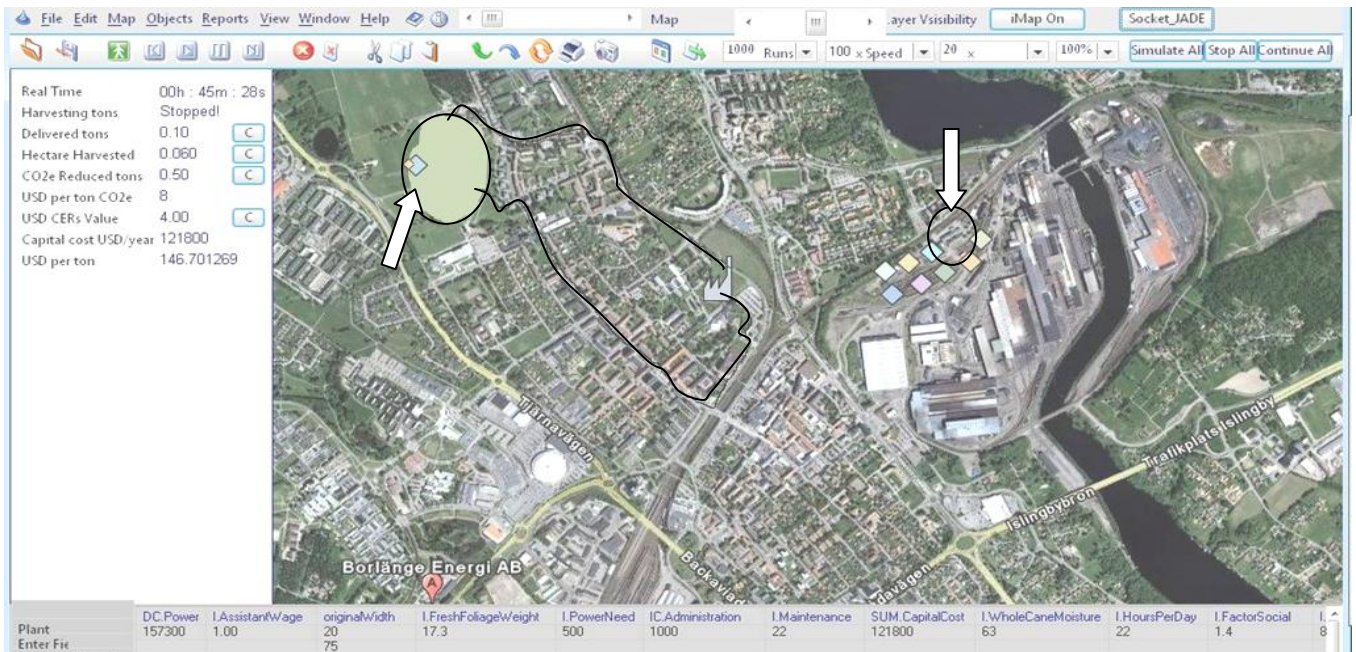


Figure 6.3 2 Runrev Revolution (BIOSIM) with one bidder process through JADE

This Figure 6.3 shows the “BioSim” interface. “BioSim” receive the bid winner information from *JADE* through Socket. As in Figure 6.2 shows that the truckcontractor4 win the bidding, so truckcontractor4 leave from their area toward the field for harvesting. All the process is working successfully.

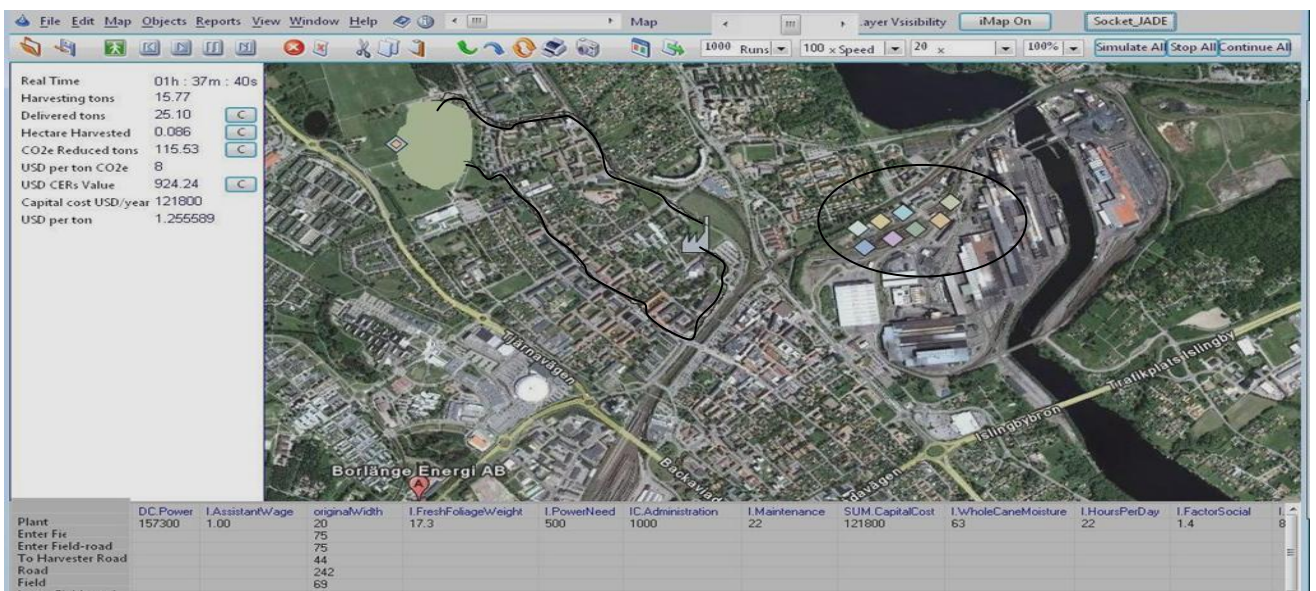



Figure 6.4 19 Runrev Revolution (BIOSIM) simulation model (truck harvesting)

This figure 6.4 shows that the truck loading () on the field.

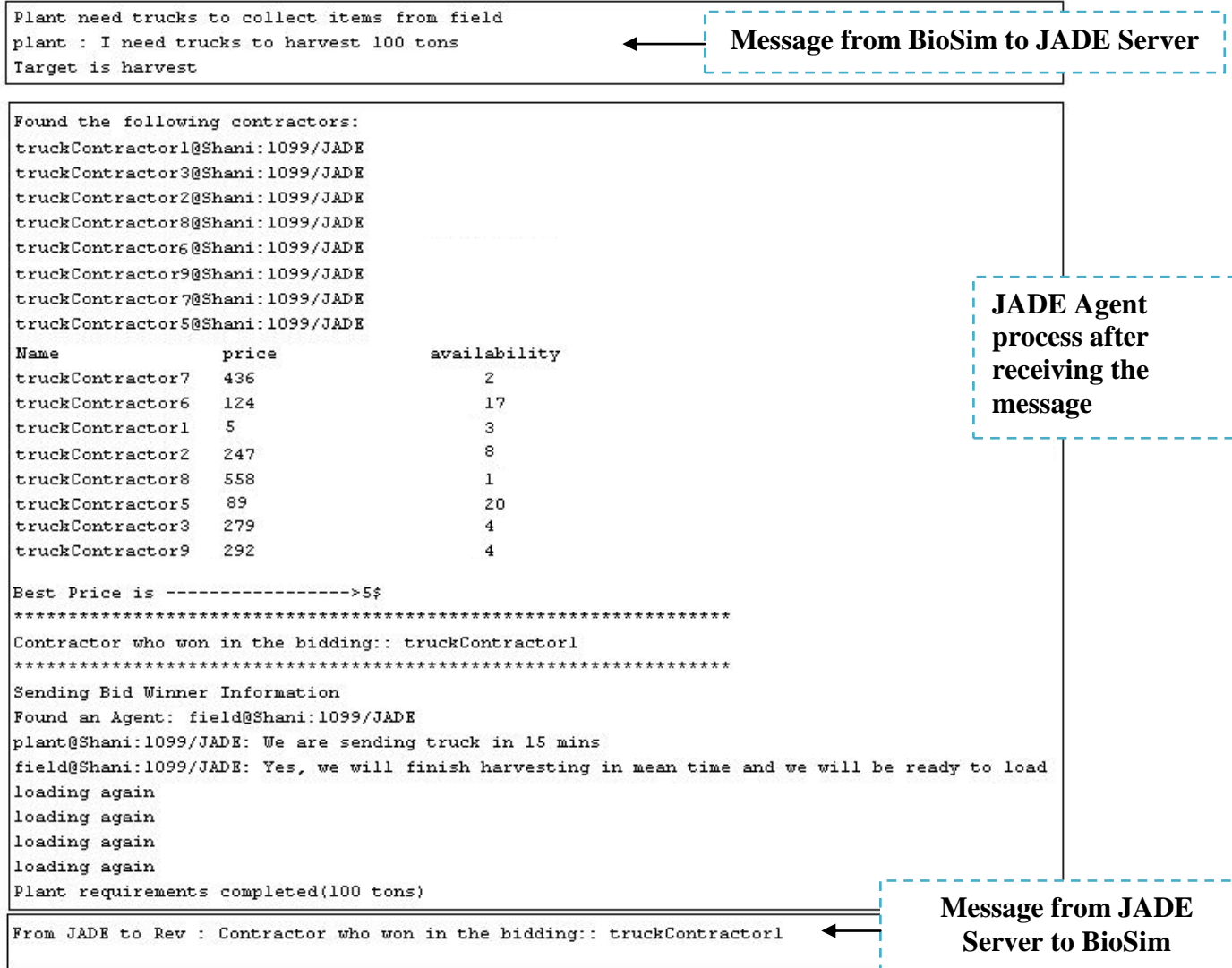


Figure 6.5 JADE bidding processes for next contractor (1field, 1plant, and n-number trucks)

This Figure 6.5 shows that the next message on JADE Server, after receiving the message from BioSim on JADE Server. JADE Server work like if then rule, it process the bidding through plant Agnet, as after the process truckcontractor1 won the bid among the other. JADE Sever send the bid winner information to BioSim.

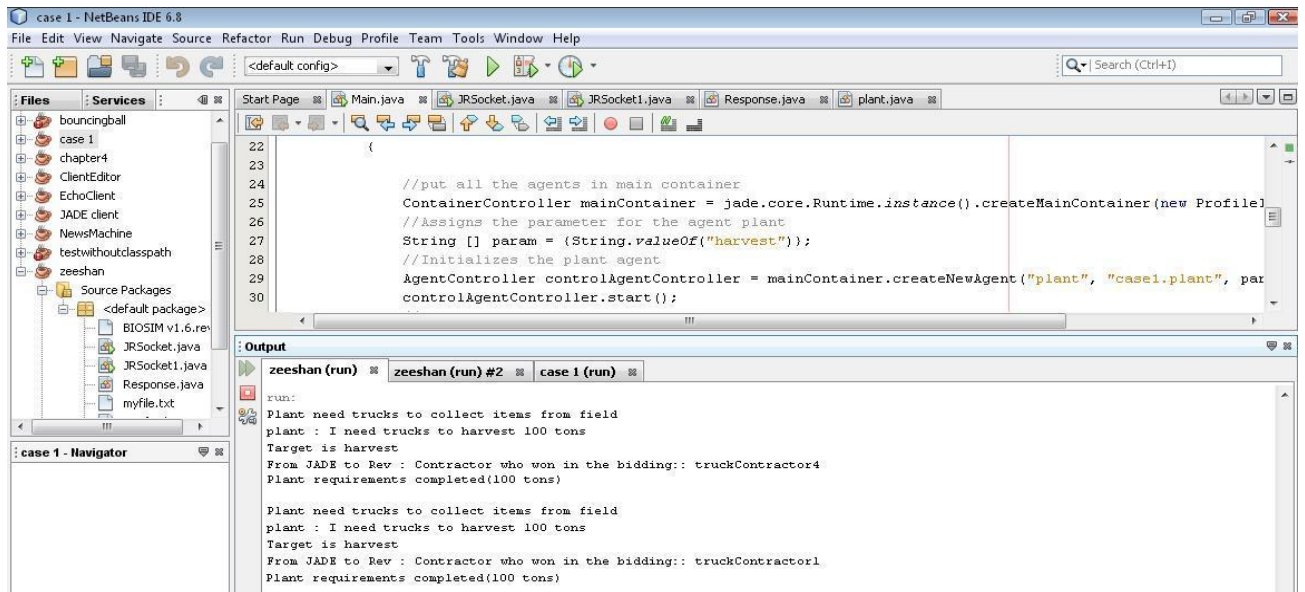


Figure 6.6 21 JADE Server process for socket connectivity

This Figure 6.6 shows that the process/results, how and what messages it read/receive from “BioSim” through socket and what kind of information is send back to “BioSim”.

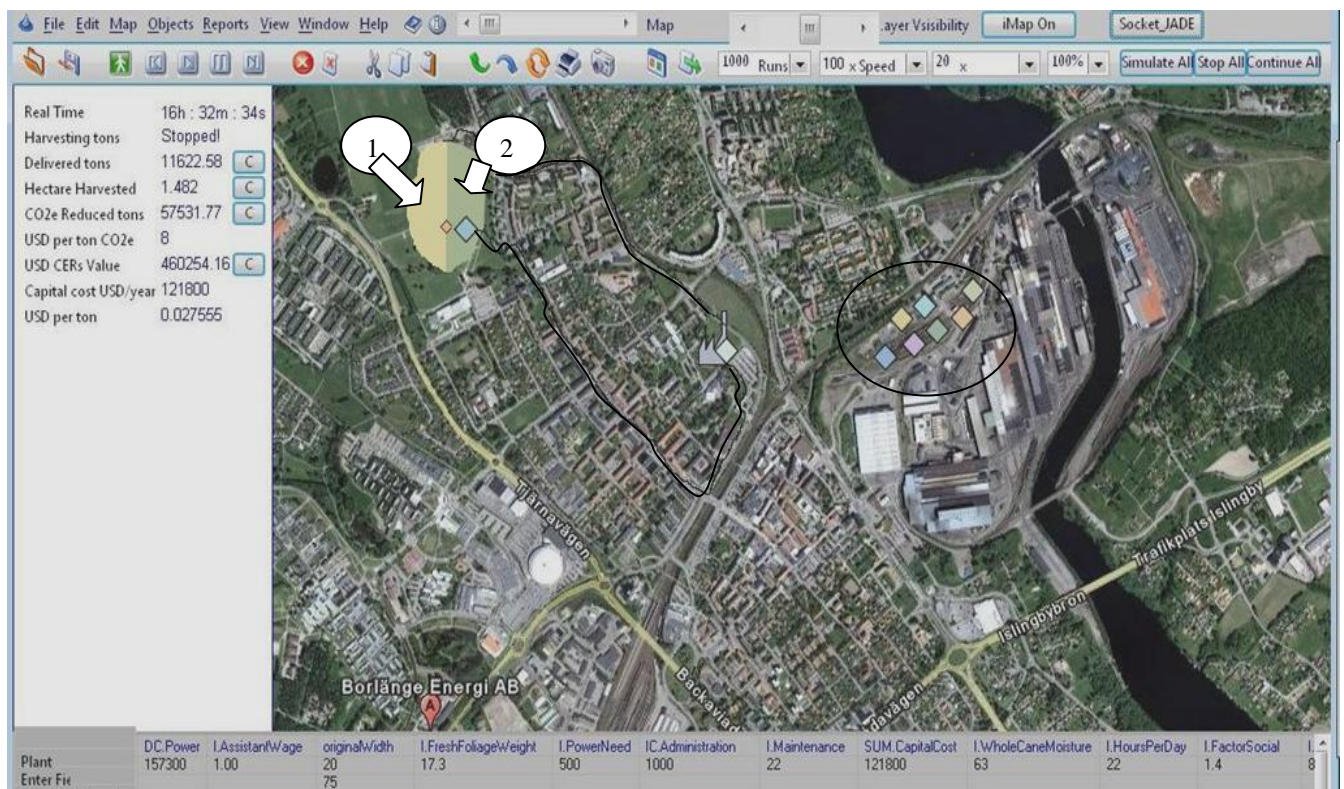


Figure 6.7 22 Runrev Revolution (BIOSIM) harvesting process

This figure shows the activity of trucks on the field and plant. (1) Shows the empty area which is completed and (2) shows the remaining raw materials on the field for harvesting. Side bar shows the status of (real time, delivered ton, cost, etc)



6.1 Call/process the JADE directly from Runrev Revolution (BIOSIM):

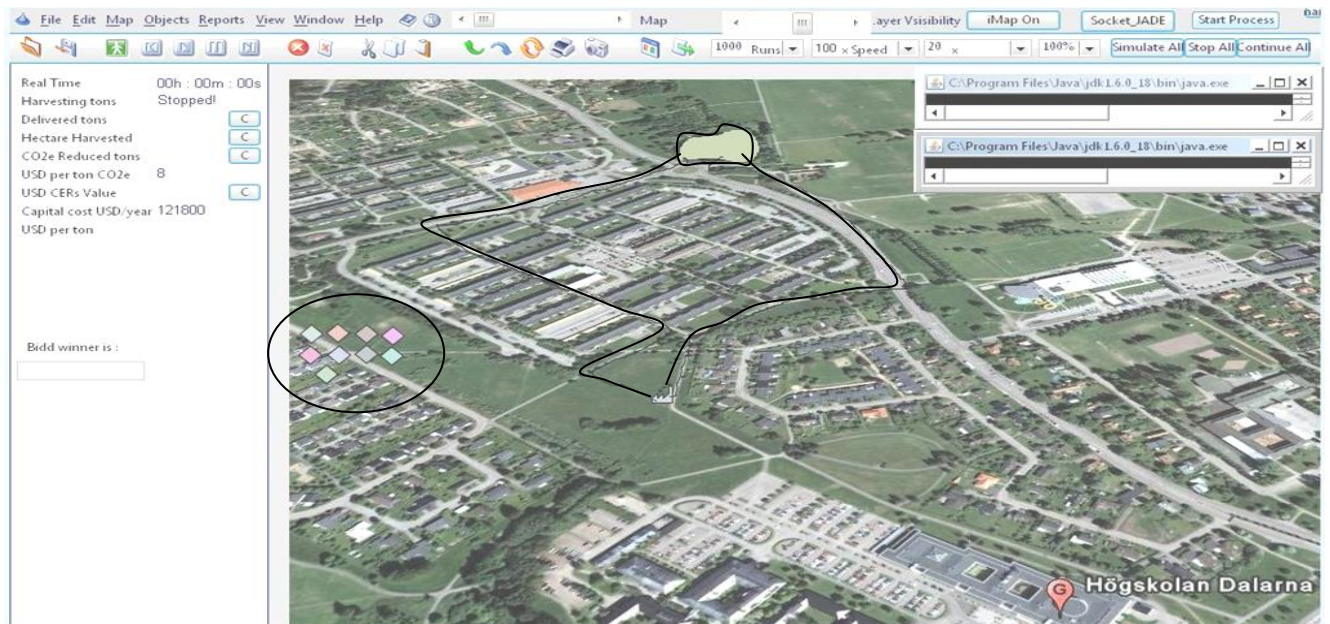


Figure 6.8 Runrev Revolution (BIOSIM)

This figure shows the connectivity with the *JADE* using BioSim (*Runrev Revolution*) through shell functions. Start process button start the process, it loads the java classes and run these files as it runs through command prompt.

```

May 24, 2010 11:33:52 PM jade.core.Runtime beginContainer
INFO: This is JADE 3.7 - revision 6154 of 2009/07/01 17:34:15
downloaded in Open Source, under LGPL restrictions,
at http://jade.tilab.com/
May 24, 2010 11:33:54 PM jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
May 24, 2010 11:33:54 PM jade.core.BaseService init
INFO: Service jade.core.messaging.MessagingService initialized
May 24, 2010 11:33:54 PM jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
May 24, 2010 11:33:54 PM jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
May 24, 2010 11:33:54 PM jade.core.messaging.MessagingService clearCachedSlice
INFO: Clearing cache
May 24, 2010 11:33:56 PM jade.mtp.http.HTTPServer <init>
INFO: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParser
impl$JAXPSAXParser
May 24, 2010 11:33:56 PM jade.core.messaging.MessagingService boot
INFO: MTP addresses
http://130.243.60.227:7778/acc
May 24, 2010 11:33:56 PM jade.core.AgentContainerImpl joinPlatform
INFO: Agent container Main-Container@Shani is ready.
Hello! Buyer-agent plant@Shani:1099/JADE is ready.
Plant need trucks to collect items from field
plant: I need trucks to harvest 100 tons
Target is harvest
Found the following contractors:
truckContractor1@Shani:1099/JADE
truckContractor2@Shani:1099/JADE
truckContractor3@Shani:1099/JADE
Name price availability
truckContractor1 656 16
truckContractor2 367 18
truckContractor3 60 12
Best Price is ----->60$
Contractor who won in the bidding:: truckContractor3
Connected to server
Found an Agent: field@Shani:1099/JADE
plant@Shani:1099/JADE: We are sending truck in 15 mins
field@Shani:1099/JADE: Yes, we will finish harvesting in mean time and we will b
e ready to load
loading again
loading again
loading again
loading again
Plant requirements completed<100 tons>

```

Figure 6.9 JADE bidding process through shell

Figure 6.9 shows the bidding process of *JADE*, after finalizing the bid winner information, *JADE* send this information to BioSim (*Runrev Revolution*) through TCP/IP.

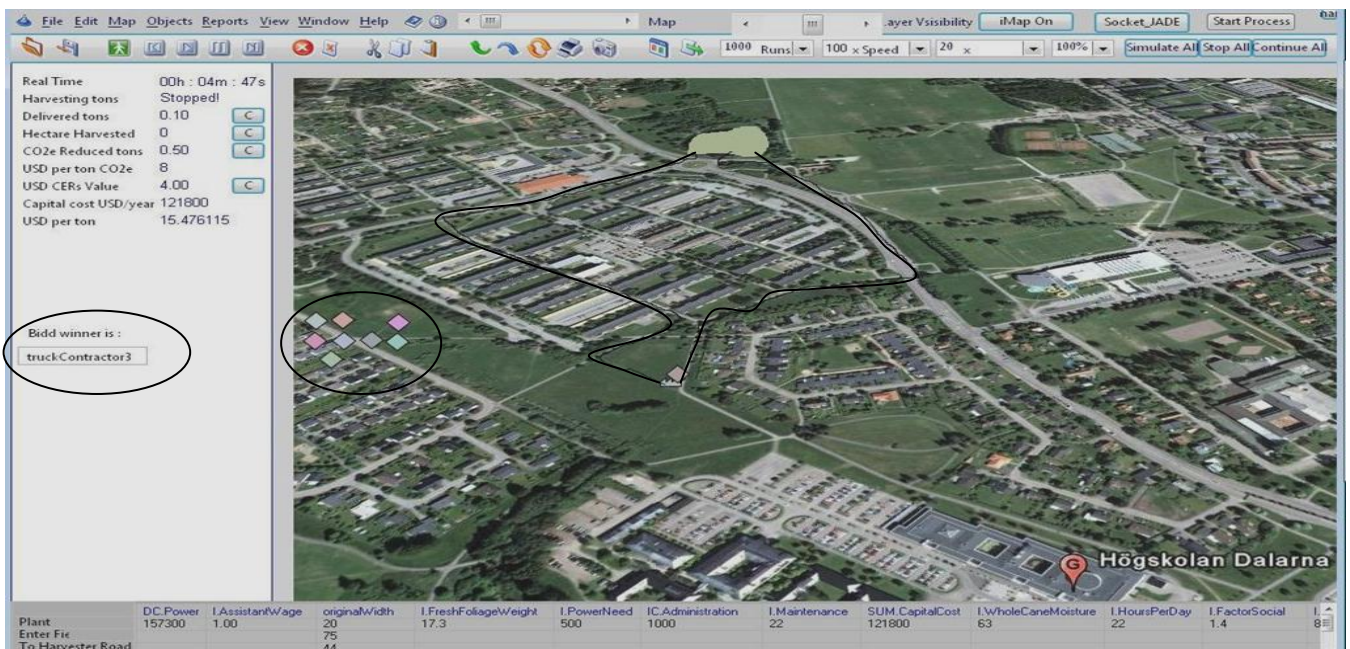


Figure 6.10 25 Runrev Revolution (BIOSIM) simulation model with bid winner information

The figure 6.10 shows the connectivity with *JADE* through Socket. It receives the result from *JADE* as per the plant requirement of bidding from the BioSim (*Runrev revolution*). As it shows that it receives the bid winner information from *JADE* which is displayed on side bar that truckcontractor 3 won the bid and it leaves their area toward the plant to fulfil the plant requirement.

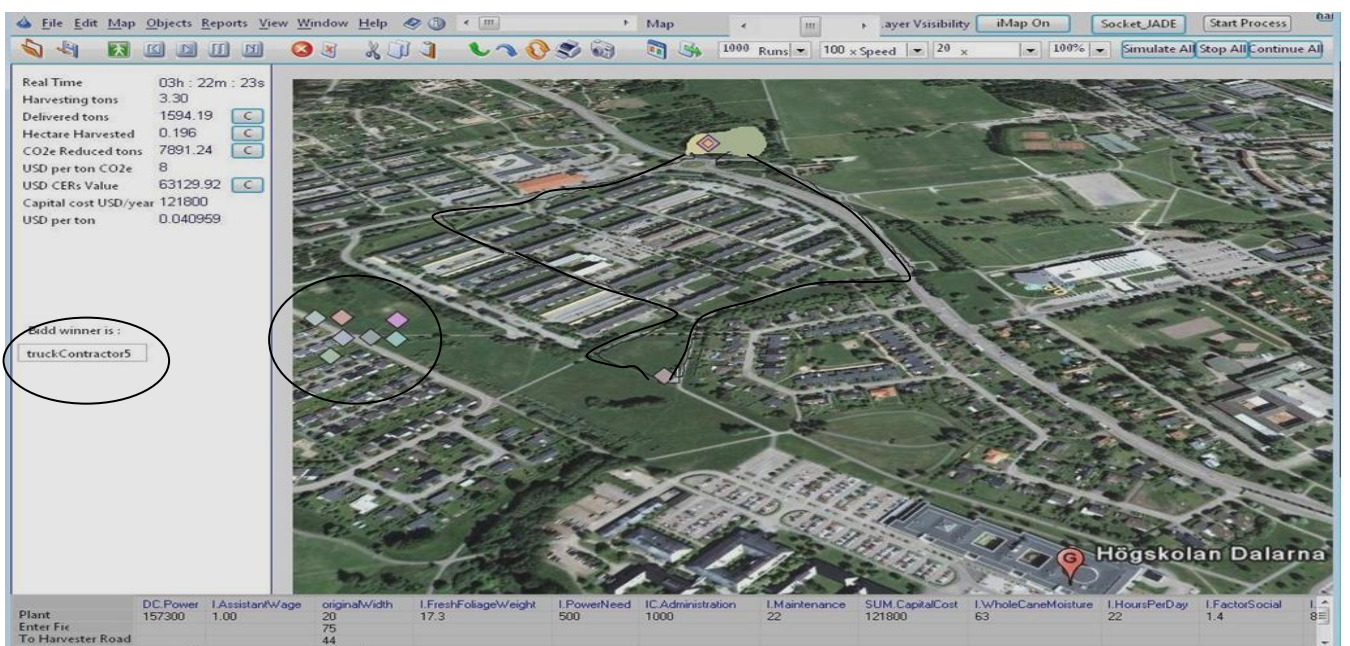


Figure 6.11 26 Runrev Revolution (BIOSIM) harvesting process



The figure 6.11 shows that if plant need more bidding (contractor) for harvesting it again call to the JADE for bidding and receive the next bid winner from the JADE. Next bidder information is shown on the side bar (“truckcontractor5”). Now both the truckcontractor3 and truckcontractor 5 are harvesting.

6.2 Connectivity through TCP/IP

JADE Server

JADE Server is created on the network system, which is connected with the other system on the same network. It is connected with the client and send/receive the messages. The figure shows the server process. This includes the requests generated by client and their response from Server. The specs of Server are (Intel(R) Pentium (R) Dual T3400 @ 2.16GHZ 2.17GHZ, 3:00 GB, Window Vista Home Premium, Service pack2)

```

Output
zeeshan (run)  zeeshan (run) #2  case 1 (run)
run:
Plant need trucks to collect items from field
plant : I need trucks to harvest 100 tons
Target is harvest
From JADE to Rev : Contractor who won in the bidding:: truckContractor1
Plant need trucks to collect items from field
plant : I need trucks to harvest 100 tons
Target is harvest
From JADE to Rev : Contractor who won in the bidding:: truckContractor1
Plant need trucks to collect items from field
plant : I need trucks to harvest 100 tons
Target is harvest
From JADE to Rev : Contractor who won in the bidding:: truckContractor5

Output
zeeshan (run)  zeeshan (run) #2  case 1 (run)
plant:I need trucks to harvest 100 tons
Target is harvest
Found the following contractors:
truckContractor6@Shani:1099/JADE
truckContractor4@Shani:1099/JADE
truckContractor1@Shani:1099/JADE
truckContractor3@Shani:1099/JADE
truckContractor2@Shani:1099/JADE
truckContractor5@Shani:1099/JADE
Name      price      availability
truckContractor6  48          13
truckContractor3  697         22
truckContractor4  592         20
truckContractor2  328         0
truckContractor1  693         18
truckContractor5  40          4

Best Price is ----->40$
*****
Contractor who won in the bidding:: truckContractor5
*****
Connected to server
Found an Agent: field@Shani:1099/JADE
plant@Shani:1099/JADE: We are sending truck in 15 mins
field@Shani:1099/JADE: Yes, we will finish harvesting in mean time and we will be ready to load
loading again
loading again
loading again
loading again
Plant requirements completed(100 tons)

```

Figure 6.12 JADE server on Network system



Client (Runrev Revolution):

The client created on the same networks which make the request to server on the IP (130.243.60.6) and on port 54322, after establishing the connection it send/receive the message from server. The specs of the Client are (CPU: 800 A, Pentium 3 LV, RAM: 256 MB).

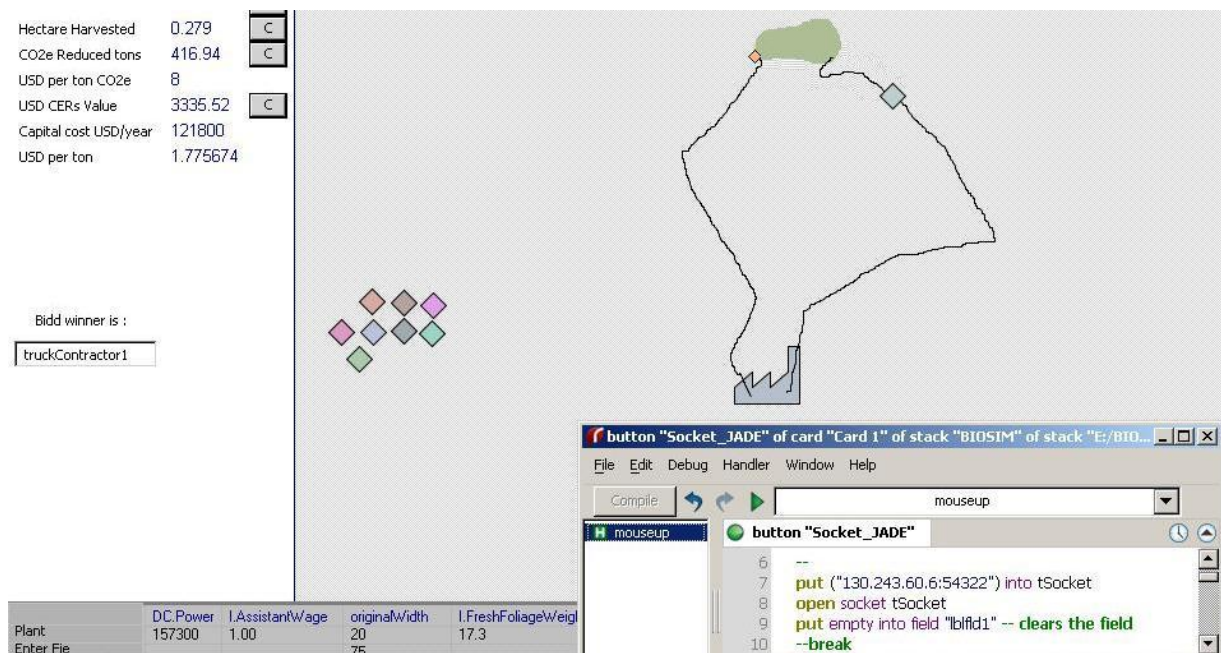


Figure 6.13 28 Runrev Revolution as a client on network system

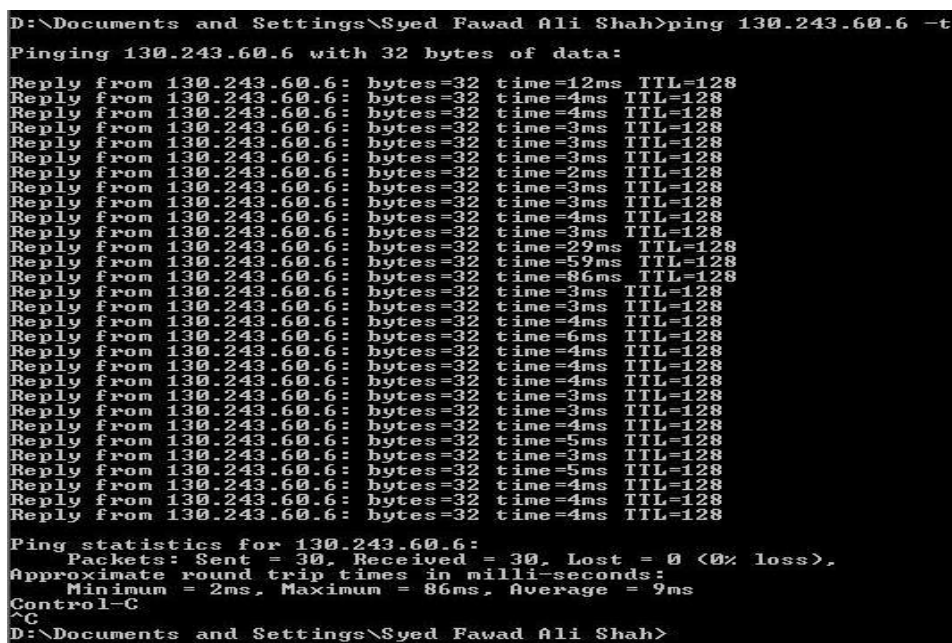


Figure 6.14 29 checking the delays on the network system



Client (Runrev Revolution):

The same process is tested on a different machine over the same network. client make the request to server on the IP (130.243.60.6) and on port 54322, after establishing the connection it send/receive the message from server. The specs of the client are (Intel(R) core (TM)2 Duo CPU T7250 @ 2.00GHZ 2.00GHZ, 2:00 GB, 32-bit, Window 7 professional)

```

Reply from 130.243.60.6: bytes=32 time=2ms TTL=128
Reply from 130.243.60.6: bytes=32 time=7ms TTL=128
Reply from 130.243.60.6: bytes=32 time=4ms TTL=128
Reply from 130.243.60.6: bytes=32 time=4ms TTL=128
Reply from 130.243.60.6: bytes=32 time=2ms TTL=128
Reply from 130.243.60.6: bytes=32 time=139ms TTL=128
Reply from 130.243.60.6: bytes=32 time=2ms TTL=128
Reply from 130.243.60.6: bytes=32 time=10ms TTL=128
Reply from 130.243.60.6: bytes=32 time=2ms TTL=128
Reply from 130.243.60.6: bytes=32 time=3ms TTL=128
Reply from 130.243.60.6: bytes=32 time=2ms TTL=128
Reply from 130.243.60.6: bytes=32 time=3ms TTL=128
Reply from 130.243.60.6: bytes=32 time=5ms TTL=128
Reply from 130.243.60.6: bytes=32 time=2ms TTL=128
Reply from 130.243.60.6: bytes=32 time=4ms TTL=128
Reply from 130.243.60.6: bytes=32 time=2ms TTL=128
Reply from 130.243.60.6: bytes=32 time=3ms TTL=128
Reply from 130.243.60.6: bytes=32 time=4ms TTL=128
Reply from 130.243.60.6: bytes=32 time=3ms TTL=128
Reply from 130.243.60.6: bytes=32 time=4ms TTL=128
Reply from 130.243.60.6: bytes=32 time=2ms TTL=128
Reply from 130.243.60.6: bytes=32 time=4ms TTL=128
Reply from 130.243.60.6: bytes=32 time=3ms TTL=128
Reply from 130.243.60.6: bytes=32 time=4ms TTL=128
Reply from 130.243.60.6: bytes=32 time=3ms TTL=128
Reply from 130.243.60.6: bytes=32 time=3ms TTL=128
Reply from 130.243.60.6: bytes=32 time=2ms TTL=128
Reply from 130.243.60.6: bytes=32 time=3ms TTL=128

Ping statistics for 130.243.60.6:
    Packets: Sent = 158, Received = 158, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 139ms, Average = 6ms
    
```

Figure 6.15 30 checking the delays on the network system

Type	DC.Power	I.AssistantWage	originalWidth	I.FreshFoliageWeight	I.PowerNeed	IC.Administration	I.Maintenar
Plant	157300	1.00	20	17.3	500	1000	22
Enter Field road			75				

Figure 6.16 31 Runrev Revolution as a client on network system



The whole system is tested successful over a server (1 Machine), client (Different Machines) on same Network. It shows that the connectivity established among these systems without any delay as tested on the same machine (localhost).

CHAPTER 7

Conclusion and Future Works:

7.1 Conclusion:

As logistic problem are very complex, in which logistics of "harvesting problem" are very much complicated due to large number of actors are involved, like cost, human activity, time, fuel consumption, etc. if the plant doesn't have its own trucks. They have to hire from private owners through bidding. With the "BioSim" simulation model for the "Harvesting problem" of logistics system the user can see all the available features. They can optimise the trucks and routs in advance through simulation.

In *Runrev Revolution* script language is very user friendly and it has advance graphical features. As *Runrev Revolution* is a Script language, which is quite slow. In this scenario/case "BioSim" (*Runrev Revolution*) is linked to *JADE* Multi Agents through Socket programming using TCP/IP to add intelligent behaviour in objects, through this communication channel the agents behaviour are send to the graphical objects and graphical objects behave like intelligent agents.

According to the observation for obtaining the best results the integration of *Runrev Revolution* (Graphical Engine), *JADE* (Multi Agents) has provided the best optimal solution for Harvesting logistics problem.



7.2 Future Work:

This project only tested the 1plant, 1field and n-number of trucks for test. So all the n-fields and n-plants and n-bidders, the distance between the field and plant, and the shortest distance should be consider as future work.

Runrev Revolution can communicate with C/C++ though external it taken the DLL files from C++ and remove C++ from itself after updating DLL, and *Runrev Revolution* can communicate with *JADE* through TCP/IP, so there is no external for java/*JADE* in *Runrev Revolution*. So it is better to develop any program which works like external for Java/*JADE*. The external start JavaVM instance within the *Runrev Revolution* process, and then it acts as a 'bridge' between *Runrev Revolution* scripts and Java classes, so that java libraries can be call from *Runrev Revolution* Script. This way, they can write their 'external' once in Java, instead of having to write and compile it for MacOSX, Windows and Linux separately. Other for future work as given below.

- Calculate the distance between the field and plant.
- When the Agent kill after finish their work in *JADE* (n-field, n-plants, n-trucks), same in the BIOSIM the truck, field or plant disappear after completing their task.
- Compare the each truck status (fuel, time, cost, etc) of harvesting for more efficiency with BIOSIM and *JADE*.
- Add the label box or in table field at the BIOSIM which shows the distance between field and plant.
- Add the map (city map or country map) in the BIOSIM and after that, placing the plant, field and truck on it, it shows the actual distance between those areas, the area may be calculated in *JADE* side. It is tested that *Runrev Revolution* can send the X-axis and Y-axis coordinates to *JADE*.

(Like one plant is placed on “ABC” location and field is on “XYZ” location, if the actual distance between these two are 30km then it should calculate and show the same)

- This simulation is calculating the advance planning. Adding the GPS services with this, it shows the live status on the actual map of their area. (Like “Vehicle Tracking” if any truck stuck on the road due to traffic blockage. It sends their status so that no more truck follow that route until the route is not clear)



References:

- [1] N. P. Davies, Q. H. Mehdi and N. E. Gough, "Toward Interfacing BDI With 3D Graphics Engines"
[http://wlv.openrepository.com/wlv/bitstream/2436/34132/2/CGAIMS_05\(a\)_Davies%20et%20al.pdf](http://wlv.openrepository.com/wlv/bitstream/2436/34132/2/CGAIMS_05(a)_Davies%20et%20al.pdf) (Access on May 2010)
- [2] Logistix Partners Oy, Helsinki, Finland, 1996 <http://www.logisticsworld.com/logistics.htm> (Access on April 2010)
- [3] Nicoleta Neagu, Klaus Dorer, Monique Calisti, Solving Distributed Delivery Problems with Agent-Based Technologies and Constraint Satisfaction Techniques.
<http://www.aaai.org/Papers/Symposia/Spring/2006/SS-06-04/SS06-04-025.pdf> (Access on April 2010)
- [4] Dan Shafer, Review: Runtime Revolution, Programming for Mere Mortals, Volume Number: 20 (2004) Issue Number: 5
<http://www.mactech.com/articles/mactech/Vol.20/20.05/RuntimeRevolution/index.html> (Access on May 2010)
- [5] The rev platform, <http://www.runrev.com/products/the-rev-platform/overview> (Access on May 2010)
- [6] Fabio Bellifemine, Agostino Poggi, Giovanni Rimassa: Developing Multi-agent Systems with JADE, (<http://www.dia.fi.upm.es/phernan/AgentesInteligentes/referencias/bellifemine01.pdf>) (Access on April 2010)
- [7] M. J. Wooldridge and N. R. Jennings, Intelligent agents: Theories, Architectures, and Languages, 2nd ed., Germany: Springer-Verlag, 1995.
<http://publishing.eur.nl/ir/repub/asset/16208/EPS2009177LIS9058922168Moonen.pdf> (Access on April 2010)
- [8] J. Ferber, Multi-agent Systems: An introduction to distributed artificial intelligence, New York : Addison Wesley Longman, 1999.
- [10] Reticular Systems. Agent Construction Tools. 2010. Available at <http://www.agentbuilder.com>. (Access on April 2010)
- [11] What Is a Socket? <http://java.sun.com/docs/books/tutorial/networking/sockets> (Access on May 2010)
- [12] [http://www.digilife.be/quickreferences/pt/java sockets.pdf](http://www.digilife.be/quickreferences/pt/java%20sockets.pdf) (Access on May 2010)
- [13] <http://cs.gmu.edu/~setia/cs707/slides/sockets.pdf> (Access on May 2010)
- [14] Home / Products / The Rev Platform / Competitive Comparison
, <http://www.runrev.com/products/the-rev-platform/competitive-comparison/> (Access on May 2010)
- <http://earth.google.com> (Google Earth 4.2.0205.5730 Server: kh.google.com) (May 2010)
(Used for map in the BioSim)

BOOKS

- Kenneth L. Calvert, Michael J. Donahoo, “TCP/IP Sockets in Java, Practical Guide for Programmers”, Second Edition, Morgan Kaufmann publications, 2008
- Bellifemine, Giovanni Caire, Tiziana Trucco, Giovanni Rimassa, “JADE PROGRAMMER’S GUIDE”, 2001
- <http://www.ryerson.ca/~dgrimsha/courses/cps720/Resources/JADE/programmersguide.pdf> (Access on April 2010)
- REVOLUTION user Guide 4.0 document is revision 16 (2009.07.15).
- <http://downloads.runrev.com/userguide/userguide.pdf> (Accessed on April 2010)
- Giovanni Caire, JADE TUTORIAL, “JADE PROGRAMMING FOR BEGINNERS”, JADE 3.7, last update: 30 June 2009
- <http://jade.cselt.it/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf> (Accessed on March 2010)
- Barry Burd, “Beginning Programming with Java™ For Dummies”, 2nd Edition, Published by: Wiley Publishing, Inc, 2005
- Harvey M. Deitel, “Java How to Program” Edition: 7th, Deitel & Deitel, Publisher: Prentice Hall, 2006
- Fabio Bellifemine, Giovanni Caire, Dominic Greenwood, “Developing Multi-Agent Systems with JADE”, Series: Wiley Series in Agent Technology , Published : 2007

Appendix A

Dictionary

An alphabetical list of explained and abbreviation used in this thesis.

AI	Artificial Intelligence
CS	Computer System
DLL	Dynamic Link Library
IP	Internet Protocol
IDE	Integrated Development Environment
JADE	Java Agent Development Environment.
MAS	Multi-agent systems
RR	Runrev Revolution
TCP	Transport Control Protocol

SOFTWARE USED: RunRev Revolution studio, NetBeans IDE 6.8 for JADE (MAS)

Runrev Revolution: Revolution is a software and use scripting language.

www.runrev.com (Feb2010)

JADE: JADE is an open source it use the java libraries for Multi Agents .

<http://jade.tilab.com/> (Feb 2010)

NetBeans IDE 6.8: <http://netbeans.org/> (Feb 2010)