Enhancing The Role Of Mobile Agent In Establishing Communication Between Local Networks

By
Nabeel Mustafa Mohammad Alassaf

Supervisor
Dr. Emad Khalid Salah
Assistant Prof.

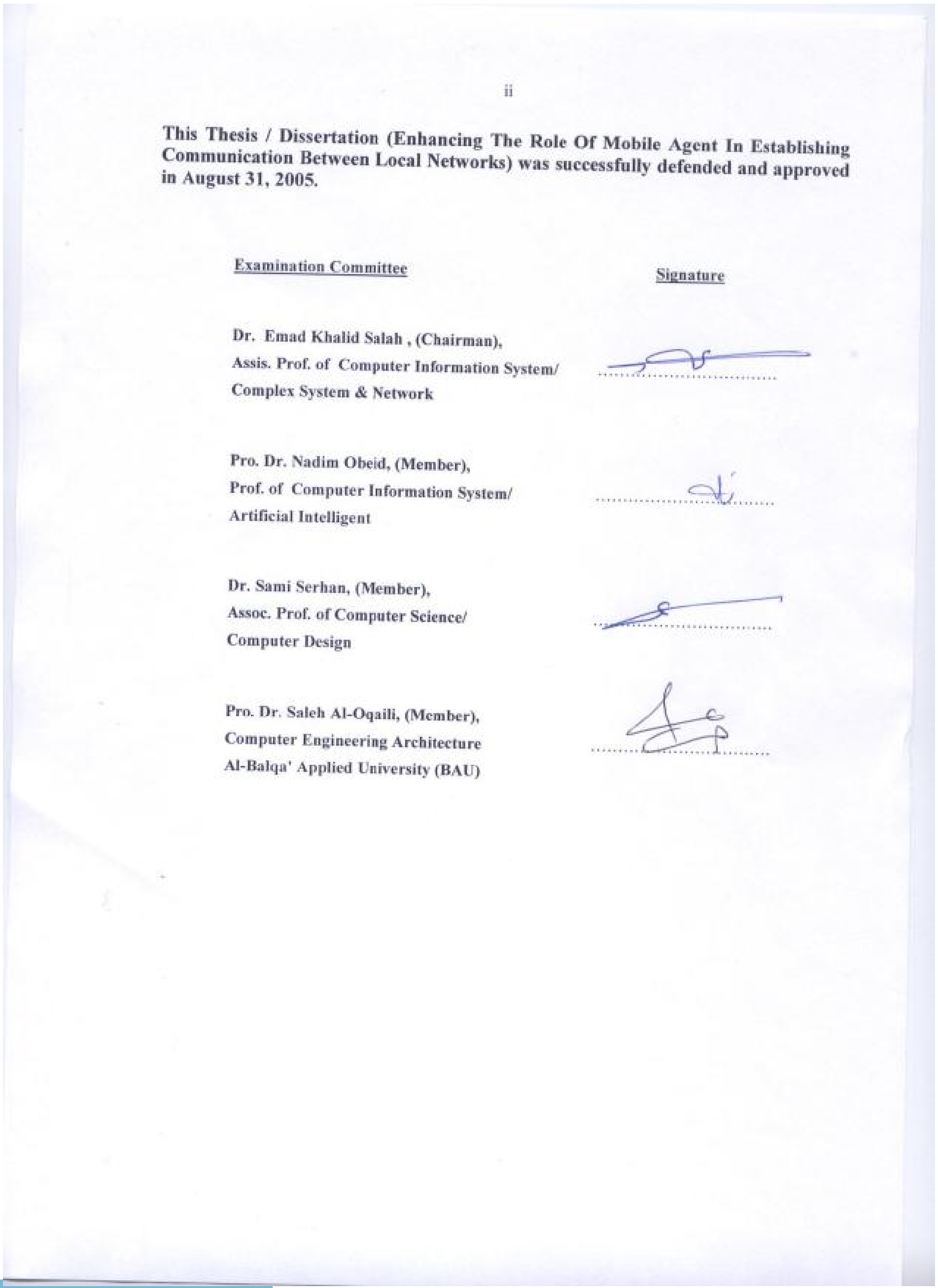
This Thesis Was Submitted In Partial Fulfillment Of The Requirements For The Master's Degree

Of Science In Computer Science

Faculty Of Graduate Studies

The University Of Jordan

August, 2005



This Thesis / Dissertation (Enhancing The Role Of Mobile Agent In Establishing Communication Between Local Networks) was successfully defended and approved in August 31, 2005.

Examination Committee	**Signature**
Dr. Emad Khalid Salah , (Chairman), **Assis. Prof. of Computer Information System/** **Complex System & Network**	[illegible]
Pro. Dr. Nadim Obeid, (Member), **Prof. of Computer Information System/** **Artificial Intelligent**	[illegible]
Dr. Sami Serhan, (Member), **Assoc. Prof. of Computer Science/** **Computer Design**	[illegible]
Pro. Dr. Saleh Al-Oqaili, (Member), **Computer Engineering Architecture** **Al-Balqa' Applied University (BAU)**	[illegible]

DEDICATION

To my beloved parents, for their unlimited love and support.

To my beloved brothers and sisters, who never stopped helping me.

To my best friends, for their encouragement and trust.

To my colleagues in the University of Jordan.

To everyone helped me to achieve this piece of work.

ACKNOWLEDGMENT

I would like to thank my supervisor, Dr. Emad Khalid Salah, for his great support, encouragement. Nevertheless, his advices were of tremendous help to the completion of this thesis.

I am grateful to Pro. Dr. Nadim Obied, for his invaluable guidance and unlimited support during the whole process, and for introducing me to the area of Mobile agent and Multi agent systems.

List of CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
ACC	Agent Communication Channel
CORBA	Common Object Request Broker Architecture
CPU	Central processing Unit
FIPA	Foundation for Intelligent Physical Agents
IDL	Interface definition language
IIOP	the Internet Inter-ORB Protocol
Java-RMI	Java Remote Method Invocation
JVM	Java Virtual Machine
LAN	Local Area Networks
MAF	Mobile Agent Facility
MAF	Mobile Agent Framework
MAP	Mobile Agent Professional
MbD	Management by Delegation
OMG	Object Management Group
ORB	Object request broker
QOS	quality of service
RAD	Rapid application development
RPC	Remote Procedure Calls
SNM	System and Network Management

SNMP	Simple Network Management Protocol
TMN	Telecommunications management networks
WAN	Wide Area Networks

Enhancing The Role Of Mobile Agent In Establishing Communication Between Local Networks

By
Nabeel M. Alassaf

Supervisor
Dr. Emad Khalid Salah
Assistant Prof.

ABSTRAC

For several years to now, the applicability and usefulness of mobile agent technologies for distributed System and Network Management (SNM) have been recognized. One of the main points is to delegate to autonomous and possibly mobile agents the administration tasks. As such, the network and computation loads are distributed instead of centralized towards and on the manager host.

Network configurations are other important tasks which can also have some effect on the effective topology of the managed whole network. Using any mobile agent based platform for system and network management implies to first deploy the infrastructure (daemons in order to host mobile agents). Secondly, be able to program itineraries composed of system or network elements that mobile agents must visit in order to execute their management tasks.

In this thesis, we introduce more effective algorithms that help us to use the mobile agent to communicate local networks by many tools that find the IP for all computers in the same class or out it and the name of that computer that registered by it on the networks and we introduce algorithms using the mobile agent to find the all open ports for remote computer and get the status of computer in selected networks to show that the computers are online or offline to communicate with it and I introduce an

algorithms that use the mobile agent as node to control and manage a remote computer and get all properties and devices of the computer and other algorithms which are explained in details in this thesis.

Chapter 1

Introduction

1.1 Problem Overview:

This research concentrates on communication between two or more local networks. This can be realized through mobile agents that navigate as required between the different networks. The idea of this thesis is to develop a computerized technique that enhances the role of mobile agent in establishing communication between local networks. The data to be sent is messages transferred via local network as mobile agents move from one network to another. Such agent can interconnect two or more networks or they can use measurements to analyze the whole network communication performance and get all the IP of computers around to my computers through mobile agent and get all open ports and check the status of any computers in the networks if it is online or offline and control the messages between computers and manages and control any computers that the mobile agent reside in it.

An agent is characterized by what it can do, and what it actually does. Agents are entities that act on behalf of other entities. The possibilities of agent technology are boundless and can be used in a wide variety of applications in almost every sphere of life today.

1.1.1 Mobile Agents:

Mobile agents are agents that can travel across a network and perform tasks on machines that provide agent hosting capability. This allows processes: (1) to migrate from one computer to another, (2) to split into multiple instances that execute on different machines, (3) to return to their point of origin. Unlike remote procedure calls, where a process invokes procedures of a remote host, process migration allows executable code to travel and interact with databases, file systems, information services another computer and other agents. We will examine the technology that underlies mobile agents and we shall present an analysis of its uses and impacts.

The size of mobile agents depends on what they do. In swarm intelligence (White et al., 1998), the agents are very small. On the other hand, configuration or diagnostic agents might get quite big, because they need to encode complex algorithms or reasoning engines. Note however, that agents can extend their capabilities on-the-fly, on-site by downloading required code off the network. They can carry only the minimum functionality, which can grow depending on the local environment and needs. This capability is facilitated by code mobility.

1.1.2 Advantages of mobile agents:

The use of mobile agents may have advantages over other implementations of agents. This does not imply that other technologies (like remote objects) cannot be used instead, because virtually any task that can be performed with mobile agents can also be performed with stationary objects. However, the traditional solutions might be less efficient, difficult to deploy, or awkward. Table 1.1 (Green et al.) contains the areas that may benefit from appropriate use of mobile agents instead of, or in addition to, classical client/server models.

Table 1.1: Advantage of mobile agents

Possible Benefit	**Justification**
Efficiency savings	CPU consumption is limited, because a mobile agent executes only on one node at a time. Other nodes do not run an agent until needed.

Table 1.1: Advantage of mobile agents (Cont.)

Space savings	Resource consumption is limited, because a mobile agent resides only on one node at a time. In contrast, static multiple servers require duplication of functionality at every location. Mobile agents carry the functionality with them, so it does not have to be duplicated. Remote objects provide similar benefits, but the costs of the middleware might be high.
Reduction in network traffic	Code is very often smaller than data that it processes, so the transfer of mobile agents to the sources of data creates less traffic than transferring the data. Remote objects can help in some cases, but they also involve marshalling of parameters, which may be large.
Asynchronous autonomous interaction	Mobile agents can be delegated to perform certain tasks even if the delegating entity does not remain active.
Interaction with real-time systems	Installing a mobile agent close to a real-time system may prevent delays caused by network congestion. In Network Management systems NM agents usually reside close to the hardware, so this advantage might not be as clear as others.

Table 1.1: Advantage of mobile agents (Cont.)

Robustness and fault tolerance	If a distributed system starts to malfunction, then mobile agents can be used to increase availability of certain services in the concerned areas. For example, the density of fault detecting or repairing agents can be increased. Some kind of meta level management of agents is required to ensure that the agent-based system fulfills its purpose.
Support for heterogeneous environments	Mobile agents are separated from the hosts by the mobility framework. If the framework is in place, agents can target any system. The costs of running a Java Virtual Machine (JVM) on a device are decreasing. Java chips will probably dominate in the future, but the underlying technology is also evolving in the direction of ever-smaller footprints.
On-line extensibility of services	Mobile agents can be used to extend capabilities of applications, for example, providing services. This allows for building systems that are extremely flexible.

Table 1.1: Advantage of mobile agents (Cont.)

Convenient development paradigm	Creating distributed systems based on mobile agents is relatively easy. The difficult part is the mobility framework, but when it is in place, then creating applications is facilitated. High level, rapid application development (RAD) environments for agents will be needed when the field matures. It is quite probable that the flourishing tools for object-oriented programming will evolve into agent-oriented development environments, which will include some functionality to facilitate agent mobility.
Easy software upgrades	A mobile agent can be exchanged virtually at will. In contrast, swapping functionality of servers is complicated; especially, if we want to maintain the appropriate level of quality of service (QoS).

1.1.3 Motivation:

Agents make an interesting topic of study because they draw on and integrate many diverse disciplines of computer science, including objects and distributed object architectures, adaptive learning systems, artificial intelligence, expert systems, genetic algorithms, distributed processing, distributed algorithms, collaborative online social environments, and security just to name a few.

Agent technology is significant because of the sustained commercial interest surrounding it. You have most likely heard of General Magic and Telescript, and maybe even IBM's Aglets Workbench (now called IBM Aglets SDK) and Mitsubishi's Concordia. Agent technology may not have hit prime time quite yet, but it does seem to be gathering its share of investment money.

Agent technology is also interesting for its potential to solve some nagging productivity problems that pester almost all modern computer users. Many agents are meant to be used as intelligent electronic gophers automated errand boys. Tell them what you want them to do search the Internet for information on a topic, or assemble and order a computer according to your desired specifications -- and they'll do it and let you know when they've finished.

1.1.4 The problems that agents solve:

Agent technology solves, or promises to solve, several problems on different fronts. Mobile agents solve the nagging client/server network bandwidth problem. Network bandwidth in a distributed application is a valuable (and sometimes scarce) resource. A transaction or query between a client and the server may require many round trips over the wire to complete. Each trip creates network traffic and consumes bandwidth. In a system with many clients and/or many transactions, the total bandwidth requirements may exceed available bandwidth, resulting in poor performance for the application as a whole. By creating an agent to handle the query or transaction, and sending the agent from the client to the server, network bandwidth consumption is reduced. So instead of intermediate results and information passing over the wire, only the agent need be sent.

Here's a related situation. In the design of traditional client/server architecture, the architect spells out the roles of the client and server pieces very precisely up front, at design time. The architect makes decisions about where a particular piece of functionality will reside based on network bandwidth constraints (remember the previous problem), network traffic, transaction volume, number of clients and servers, and many other factors. If these estimates are wrong, or the architect makes bad decisions, the performance of the application will suffer. Unfortunately, once the system has been built and the performance measured, it's often difficult or impossible to change the design and fix the problems. Architectures based on mobile agents are potentially much less susceptible to this problem. Fewer decisions must be made at design time, and the system is much more easily modified after it is built. Agent architectures that support adaptive network load balancing could do much of the redesign automatically.

Agent architectures also solve the problems created by intermittent or unreliable network connections. In most network applications today, the network connection must be alive and healthy the entire time a transaction or query is taking place. If the connection goes down, the client often must start the transaction or query from the beginning, if it can restart it at all. Agent technology allows a client to dispatch an agent handling a transaction or query into the network when the network connection is alive. The client can then go offline. The agent will handle the transaction or query on its own, and present the result back to the client when it re-establishes the connection.

Agent technology also attempts to solve (via adaptation, learning, and automation) the age-old (not to mention annoying) problem of getting a computer to do real thinking for us. It's a difficult problem. The artificial intelligence community has been battling these issues for two decades or more. The potential payoff, however, is immense.

1.2 Mobile Agent Framework (MAF):

As with any other communications-related activity, the general acceptance of mobile agents for network management activity will depend heavily upon standards. The Open Management Group (OMG) has already begun work in the area of mobile agents, and a draft standard has been tabled for discussion OMG (Cheng and Covaci, 1997). The proposed standard attempts to be platform neutral and has each chunk of mobile code identify itself with a language, or execution environment requirement. The proposal identifies the need for mobile code regions, with gateways between them that provide an agent application virtual layer on top of the actual network. This architecture is shown in Figure 1.1

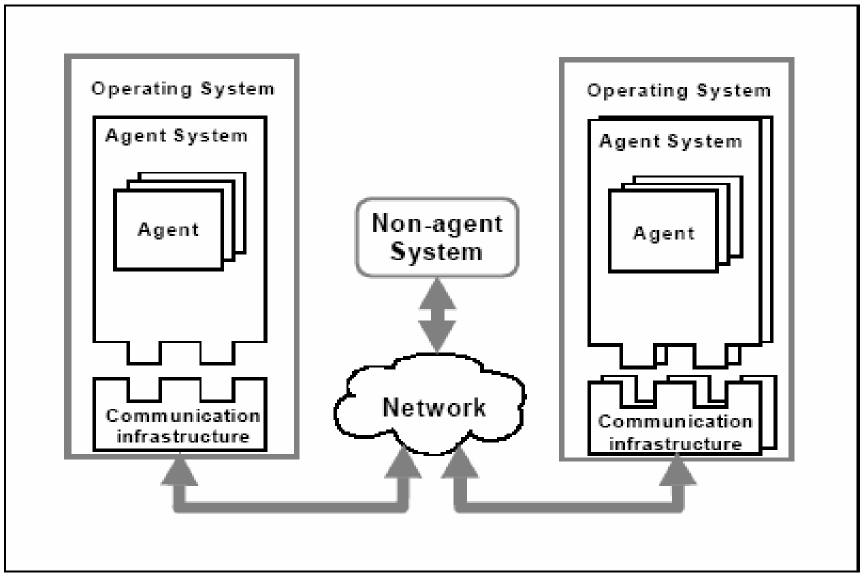


Figure 1.1: Mobile Agent Facility Architecture

An agent region is defined as a set of agent systems that can access each other, possessing similar authority and identifying a default migration pattern. Mobile agent facilities include the storage and retrieval of agents, remote agent creation transfer and agent method invocation. The draft standard also draws heavily on CORBA, with the Internet Inter-ORB Protocol (IIOP) being used as the transport protocol, and hinting that many pre-defined CORBA services, such as naming, may be used to support mobile agent activity.

Two MAF objects and their interfaces are defined in the specification; the MAF Agent System Interface and the MAF Finder Interface. The MAF Finder provides a naming service for agents, one MAF Finder (at most) being provided per region. The MAF Finder is registered as a CORBA object, the intention being that one agent, or MAF Client, may locate, and communicate with, another agent. The MAF

Agent System Interface provides standard management operations for agents, such as receive, create, suspend and terminate. The specification provides interface definition language (IDL) details on agent naming, authority (or in CORBA terms, principal) and type; together these being used to generate a globally unique agent name.

While the MAF specification is a useful starting point, it is limited in scope. No notification services are defined, security is briefly mentioned and the MAF Finder, in particular, has an impoverished interface. For example, the query, "List all agents of type X" cannot be asked.

1.3 Applications of Mobile Agents:

Mobile agents can be useful for many applications. Most obviously, **information retrieval** on the network can be supported much more efficiently if an agent representing a query can move to the place where the data are actually stored, rather than having to move all of the data across the network for sifting (and subsequently discarding most of the transmission). This works especially well for **non anticipated queries**, i. e., and the implementor of a database system cannot in general foresee everything users might want to find out and provide code to do the relevant analysis. Thus, if users have to write their own custom retrieval software, an agent-based approach can save a lot of network traffic. This is even more evident when taking into account that agents can move to other sources of information if that seems more promising. Techniques such as **semantic routing** dispatching a query according to where it is most likely to be answered, rather than according to some predetermined addressing information can be used to further boost such a system’s utility and ease of use.

Another area where mobile agents can profitably be used is **network management**. In big networks, comprising hundreds or thousands of connected computers, operations monitoring and fault detection is very difficult and involves large amounts of logging data. It is not possible to prefabricate diagnostic programs for every eventuality, but it would be feasible to use mobile agents to keep tabs on the system, home in on possible trouble spots or performance bottlenecks and bring them to the attention of the maintainers. Goldszmidt and Yemini introduce such a system based on ‘delegation agents’ (Goldszmidt and Yemini, 1995) which are dynamically linked into remote processes. As an aside, the potential of mobile agents for network management is of apparent interest to telecommunications companies, especially since current deregulation schemes keep increasing the importance of ‘quality of service’ as perceived by their customers (Lawrence, 1995).

Electronic commerce is another domain which seems amenable to mobile agents: Business on the Internet is becoming a reality, and, as standards for electronic payment are deployed, commercial ‘premises’ accessible via the net will probably mushroom. Mobile agents can help locate the cheapest offerings, negotiate deals or even conclude business transactions on behalf of their owners. Of course, the legal and commercial implications of large scale electronic commerce are daunting and cannot be discussed in this paper. Mobile agents may be a way for customers to stay on top of these developments.

Finally, an important application of mobile agents concerns **mobile computing**. Portable computers become smaller and more powerful, but wireless access to a fixed information infrastructure is likely to stay slow and cumbersome due to restrictions on radio transmission. Besides, to minimize power consumption and transmission costs, users will not want to remain on-line while some complicated query is handled on their behalf by the fixed computing resources. Mobile agents offer a promising way out of this dilemma: users simply submit mobile agents which embody their queries and log off, waiting for the agents to deposit their results ready to be picked up at a later time.

Obviously, none of these applications absolutely require the use of mobile agents most could be handled by stationary programs and some suitable communications paradigm such as RPC. However, this could only be done at a price of increased system (and network) load and, possibly, at the inconvenience of the users. Thus there is merit in researching the details of mobile agents for these applications.

It is also important to point out that agents are not forced to move, even though the system may allow them to do so. Some agents may be too big to move comfortably, and for others there may be no necessity. Such stationary agents could still communicate with their mobile counterparts to take part in an agent-based system.

1.4 Thesis Outline

The thesis is organized as follows: Chapter two presents related work and the frame work of mobile agent such as CORBA and FIPA. In Chapter three we introduce some algorithms for location independent of mobile agent and example for these algorithms. And this algorithm is helping me to manage the mobile agent when it navigates the networks to establish the communication between computers.

Implementation tools for mobile agent communication tools which are introduced with details in Chapter four and this tools are called mobile agent professional tools (MAP), including results of two different implementations of using mobile agent, first using mobile agent for control remote computer and second using mobile agent not for control remote computer but to get global information about all computer in the same domain of my device as <IP, Computer Name> this is done by mobile agent professional tools (MAP). Chapter four also introduces the all algorithms that we write and follow it to build the mobile agent professional (MAP) tools and get the full area communication between computers using mobile agent.

Finally, cling remarks summarizing the main advantages and limitations of the proposed techniques, with directions for possible future work in the field of this thesis, are presented in the chapter five.

Chapter 2

Mobile Agent Communication

2.1 Introduction:

Mobile agents have become a very popular research topic lately. Many technologies facilitate moving objects between hosts. Agent mobility presents distinct challenges not handled by traditional, distributed and/or communication frameworks. Naturally, mobile agents change location over time, making communication targeted messages with them difficult to achieve. However, a communication infrastructure must handle the case when the agent may be migrating while the message is sent. Then we intend to build a system that connect different networks by navigating mobile agents as required between different local networks in order to that satisfy the communication requirements unique to mobile agents. The infrastructure has enabled efficient control and monitoring of our mobile agents and has to facilitate mobile agent collaboration and coordination.

2.2 The most Groups that addressing mobile agent communication:

In this section we review (1) two of the most significant open specifications address mobile agent communication. which are: the Foundation for Intelligent Physical Agents (FIPA) (FIPA, 1998) and the Object Management Group (OMG) (Glass, 1999), (2) the two most prevalent approaches to communication distribution which are (i) use of a proprietary message routing architecture and (ii) extending a distributed object system, such as CORBA (Vinoski, 1997). We will discuss the main point of these approaches as it implemented. And finally we will review some approach to manage networks such as Management by Delegation approach and others important approach and subject that related to my thesis.

However, we shall begin with the work of StraBer and Schwem (Straber and Schwem, 1997) where they have contrasted the performance of mobile agent solutions with Remote Procedure Calls (RPC). An RPC allows a procedure to be executed on a Remote server, transfer the control flow and some arguments, from the client to the server, until the request is executed and the results are returned. The authors developed a performance model for mobile agent systems in which agents can alternatively use RPC or agent migration to interact with applications on other platforms. They concluded that the selection of RPC or agent migration depended on several factors, including network delay, throughput, migration overhead, number of messages, number of platforms involved, and code caching. The authors did not evaluate the impact of adding security to the performance model. The addition of encryption adds an additional overhead to both the RPC and the mobile agent communication cost. In RPC, as the number of messages increases, so does the overhead cost of encrypting them. In contrast, a mobile agent would only need to be encrypted twice, once on the way to the host, and once on the way back. In the cases of lightweight agents and frequent RPC message passing, lightweight mobile agents may be cast in a better light as the overhead of encrypting a single lightweight agent should be lower than the overhead of encrypting several RPC messages. As the agent becomes more mobile and migrates more frequently, however, the agent migration overhead related to security increases, since the agent is encrypted/decrypted at each hop.

2.2.1 Foundation for Intelligent Physical Agents (FIPA):

The Foundation for Intelligent Physical Agents (FIPA) draft specification for Agent Management (FIPA, 1998) defines an Agent Communication Channel (ACC) that is responsible for routing messages among agents within the platform and to agents resident on other platforms. Two communication options are specified for agent-to-agent communication:

• Agents can request their local ACC to route messages to target agents and ACC.

• Agents can contact the ACC of target platforms directly responsible for routing messages to target agents. ACC support for agent mobility is only optional at this stage of the specification, and the specification does not address the means of delivery.

2.2.2 Mobile Agent Facility (MAF):

OMG has published a Mobile Agent Facility (MAF) specification (Glass, 1999) as a CORBA Facility. The objective of the specification is to promote a standard interface to diverse mobile agent architectures. The specific issues addressed are agent management, tracking, and transport. Interestingly, agent communication is declared outside the scope of the MAF specification due to CORBA's extensive coverage of object communication. This implies that CORBA's communication framework is thought to be sufficient to support mobile agent communication, but the specification later clearly acknowledges that current distributed objects systems do not meet the communication requirements of mobile agents. Both the MAF and FIPA specifications focus exclusively on unicast or one-to-one communication. The MAF specification is constrained by the CORBA communication framework, while the FIPA specification appears more flexible; Multicast messaging could be an optional feature of the ACC.

2.2.3 Jumping Beans:

Jumping Beans [online] (1999) builds on the Java platform and provides a framework for Java programs to “jump” from computer to computer. The Jumping Beans architecture is based on client-server architecture, where programs moving from one host to another must do this through a central management server. The central management server has a very strict security system. By using a central management server, the Jumping Bean framework is best fitted for small distributed net works rather than WANs.

2.2.4 Novel approach:

Liotta et al. (2002) propose a novel approach to evaluating complex mobile agent systems based on a hybrid framework which allows the execution of prototype agent code over simulated internet-works. In this way it is possible to realize arbitrarily complex mobile agent systems and evaluate them over arbitrarily complex inter-networks, relying on full support to physical, link, network and transport layers for fixed and mobile networks. And they build networks using a mobile agent using simulator that travel from one node to another using the shortest path in the networks end with the full networks structure.

2.2.5 Conventional approach:

Liotta et al. (2002) Argue that the conventional approach to network monitoring based on either *management protocols* or *distributed object* technologies, cannot fully satisfy the requirements of future networked systems explained above. Intuitively, in order to monitor large-scale dynamic systems we need a distributed monitoring model that adapts to the monitored System. After discussing the limitations of monitoring models in the Simple Network Management Protocol (SNMP), telecommunications management networks (TMN), and distributed object approaches such as the Common Object Request Broker Architecture (CORBA) and Java Remote Method Invocation (Java-RMI), we discuss the potential of employing mobile agents in future network monitoring systems. A mobile agent is essentially an autonomous object containing the logic to perform a given task and possibly migrate under its own control from node to node in a network.

Liotta (2001) Attempt to build a monitoring system targeted for large-scale, dynamic systems. Code mobility offers a powerful means to pursue that but also poses a number of problems. The investigation of efficient algorithms to solve the agent location problem. This is an *NP*-complete problem when striving for optimality. Approximate solutions exist but do not suit the requirements of the agent location problem.

2.2.6 Management by Delegation approach:

A seminal work towards **strongly distributed hierarchical management** is the one introduced by Yemini *et al* with their Management by Delegation (MbD) framework (Yemini et al., 1991) MbD also represents one of the first concrete attempts to make use of Mobile Code in Network Management. Its inventors claim that MbD is not only a technique that allows for dynamic decentralization and automation of management functions, but it also introduces a paradigm shift in the management arena (Goldszmidt and Yemini, 1998).

The basic underlying principle of MbD is that management processing functions can be delegated dynamically to the network elements and executed locally rather than centrally. Thus, instead of moving raw data from the elements to a central management application, the application itself is moved and executed at the elements where data actually resides. An MbD platform is implemented as a set of *elastic servers* residing in the network elements. An elastic server is a multithreaded process whose *program code* and *process state* can be modified, extended and/or contracted during its execution (Goldszmidt, 1993). *Server elasticity* contrasts with the traditional *client-server* paradigm, which does not provide support for such a dynamic transfer of functionality between client and servers. Thus elastic servers introduce a new paradigm of interaction between components in distributed applications and provide a powerful mechanism to dynamically compose distributed applications by connecting and integrating independently delegated programs. A manager can dynamically dispatch *delegated agents* to remote elastic servers using a *delegation protocol* and can then control their execution. This protocol includes service primitives to delegate, instantiate, suspend, resume, abort and remove delegated programs. Thus, by using this protocol, a manager application can augment, during execution time, the functionality of a subordinate element, allowing it to perform an open-ended set of management programs. Delegated agents can monitor, analyze, and control devices independently from the manager, except where explicit coordination is required. MbD supports *dynamic delegation* (Mountzia and Dreo-Rodosek, 1996) and provides mechanisms for both *spatial* and *temporal distribution* (Goldszmidt and Yemini, 1996). Several applications have been prototyped on this MbD 41 platform (Goldszmidt, 1996) showing the advantages, and some of the fields of applicability for the delegation framework. For instance, a scenario of ATM switch management is reported in (Goldszmidt and Yemini, 1996). Another example is an application for real-time monitoring of the *health* of large-scale networks (Goldszmidt, 1996) and for the management of *stressed* networks (Meyer et al., 1995). Further, some considerations describing how MbD can reduce the *management control loop* and thus make networks more *reliable*, are reported in (Goldszmidt, 1996) and (Meyer et al., 1995) .

More generally, a class of applications which might benefit from MbD is described in (Goldszmidt, 1996) and (Meyer et al., 1995). Therefore, in this MbD framework, delegated scripts can operate independently from the manager, and on behalf of it, whilst managers are concerned with the management of delegated scripts. Thus, this platform is a feasible base-framework for autonomous, hierarchical, and delegated management.

In Bohoris (2000) they have looked at various models of agent mobility in the particular context of Network performance monitoring. We have introduced the concept of *constrained mobility* and discussed its practical use for dynamically programming network elements. In essence, agent Constrained mobility is achieved when Mobile Agent technologies are used in a constrained fashion; that is by limiting the multi-hop migration ability of the agent according to a *single-hop* migration scheme.

The agent migrates from the management station (where it is created) to a remote machine, where it executes a task and terminates upon completion. This concept is derived from MbD with the Difference that, in our case, what is shipped from source to destination is an autonomous mobile software agent; whereas an MbD ***delegated agent*** may be seen as a component aimed at dynamically enhancing the ***elastic server*** capability. In our performance monitoring system, Mobile Agents are created at the network management level according to the user requests and then migrate to network elements to perform monitoring functions in a local manner. The behavior of the monitoring algorithms can be customized, enabling dynamic programmable functionality to be provided directly in the managed network elements.

2.2.7 Publish-subscribe communication mechanism:

An important contribution for independent and autonomic behaviors of agents is the ability to communicate with other agents and software components. The use of the publish-subscribe genre, allows agents of different toolkits and other software components to communicate in a *one-to-one* or *one-to-many* mode. This alleviates existing restrictions of mobile agent communication mechanisms, which are designed to enable communication between agents only of the same platform. Having one common means of communication, in the form of a shared publish-subscribe middleware, also reduces the complexity of the environment and simplifies security barriers. Instead of having multiple communication mechanisms, consuming different channels, all communication is performed with a fixed set of dedicated ports as defined by the publish-subscribe communication mechanism (Padovitz et al., 2002).

The usefulness of visualization as illustration and communication tool has been explored as well (Holliday, 2003). They recognize that the AgentViz can be used as an illustrative and educational tool, much in the same manner as the systems described in these reports.

Loke et al. (2000) specify that Agents can dynamically change the type and nature of messages they would like to receive as well as produce different notifications based on their current context.

Voyager (Glass, 1999) developed by Object Space Co. is a product family consisting of an object request broker (ORB) and an application server supporting mobile agents.

Chapter 3

Algorithms of Location-Independent Communication between Mobile Agents

3.1 Introduction:

Mobile agents, units of executing computation that can migrate between machines, have been widely argued to be an important enabling technology for future distributed systems.

(Chess et al., 1997) introduces a new problem to ease application writing, one would like to be able to use high-level ***location independent*** communication facilities, allowing the parts of an application to interact without explicitly tracking each other's movements. To provide these above standard network technologies (which directly support only location-dependent communication) requires some distributed infrastructure.

(Pawel et al., 2000) and (Pawel ,2000), argue that the choice or design of an infrastructure must be somewhat application-specific — any given algorithm will only have satisfactory performance for some range of migration and communication behavior; the algorithms must be matched to the expected properties (and robustness and security demands) of applications and the communication medium. Some applications also demand disconnected operation (on laptops) and a higher level of fault-tolerance.

The goal of this chapter is to describe some algorithms which might be useful for building such infrastructures. These are simple, generic versions of the algorithms which are used in real distributed systems with object mobility and in mobile networks. (Pawel et al., 2000), discussed a small application, the ***Personal Mobile Assistant,*** and the design of an infrastructure suited to it.

3.2 Example Algorithms:

We discuss some algorithms and give some hints about the infrastructure Scalability and fault-tolerance; these algorithms are helps in managing the mobile agent when it navigates the networks to establish the communication between computers.

3.2.1 Central Server:

Central Forwarding Server: The server records the current site of every agent. Before migration an agent **A** informs the server and waits for ACK (containing the number of messages sent from the server to **A**). It then waits for all the messages due to arrive. After migration it tells the server it has finished moving. If **B** wants to send a message to **A**, **B** sends the message to the server, which forwards it. During migrations (after sending the ACK) the server suspends forwarding.

Central Query Server: The server records the current site of every agent. If **B** wants to send a message to **A**, **B** sends a query (containing the message ID) to the server asking for the current site of **A**, gets the current site **s** of **A** and sends the message to **s**. The name **s** can be used again for direct communication with **A**. If a message arrives at a site that does not have the recipient then a message is returned saying 'you have to ask the name of the server again'. Migration support is similar as above.

Home Server: Each site **s** has a server (one of the above) that records the current site of some agents usually these which were created on **s**. Agent names contain an address of the server which maintains their locations. On every migration, agent **A** synchronizes with the server whose name is part of **A**'s name. If **B** wants to send a message to **A**, **B** resolves

A's name and contacts **A**'s server. Other details are as above.

Discussion: when migrations are rare and in the case of stream communication or large messages, the Query Server seems the better choice. However, Central Forwarding and Query Servers do not scale. If the number of agents is growing, and communication and migration are frequent, the server can be a bottleneck. Home Servers can improve the situation. The infrastructure can work fine for small-to-medium systems, where the number of agents is small and the server is a single-point of failure. In such algorithms we can use some of the classical techniques of fault-tolerance, e.g. state check pointing, message logging and recovery. We can also replicate the server on different sites to enhance system availability and fault-tolerance.

3.2.2 Forwarding Pointers:

Algorithm: There is a forwarding daemon on each site. The daemon on site **s** maintains a guess about the current site of agents which migrated from **s**. Every agent knows the initial home site of every agent (the address is part of an agent's name). If **A** wants to migrate from **s1** to **s2** it leaves a forwarding pointer at the local daemon. Communications follow all the forwarding pointers. If there is no pointer to agent **A**, **A**'s home site is contacted. Forwarding pointers are left around forever.

Discussion: There is no synchronization between migration and communication as there was in centralized algorithms. A message may follow an agent which frequently migrates, leading to a race condition. The Forwarding Pointers algorithm is not practical for a large number of migrations to distinct sites (a chain of pointers is growing, increasing the cost of search). Some "compaction" methods can be used to collapse the chain, e.g. movement-based and search-based. In the former case, an agent would send backward a location update after performing a number of migrations; in the latter case, after receiving a number of messages (i.e. after a fixed number of "find" operations occurred).

Some heuristics can be further used such as search-update. A plausible algorithm can be as follows. On each site there is a daemon which maintains forwarding addresses (additionally to forwarding pointers) for all agents whichever visited this site. A forwarding address is a tuple (Timestamp; site) in which the site is the last known location of the agent and timestamp specifies the age of the forwarding address. Every message sent from agent **B** to **A** along the chain of forwarding pointers contains the latest available forwarding address of **A**. The receiving site may then update its forwarding address (and/or forwarding pointer) for the referenced agent, if required. Given conflicting guesses for the same agent, it is simple to determine which one is most recent using timestamps. When the message is eventually delivered to the current site of the agent, the daemon on this site will send an ACK to the daemon on the sender site, containing the current forwarding address. The received address replaces any older forwarding address but not the forwarding pointer (to allow updating of the chain of pointers during any subsequent communication). A similar algorithm has been used in Emerald (Jul et al., 1988), where the new forwarding address is piggybacked onto the reply message in the object invocation. It is sufficient to maintain the timestamp as a counter that is incremented every time the object moves.

A single site fail-stop in a chain of forwarding pointers breaks the chain. A solution is to replicate the location information in the chain on **k** consecutive sites, so that the algorithm is tolerant of a failure of up to **k-1** ad joint sites. Stale pointers should be eventually removed, either after waiting a sufficiently long time, or purged as a result of a distributed garbage collection. Distributed garbage collection would require detecting global termination of all agents that might use the pointer, therefore the technique may not always be practically useful. Alternatively, some weaker assumptions could be made and the agents decide arbitrarily about termination, purging the pointers beforehand.

3.2.3 Broadcast:

Data Broadcast: Sites know about the agents that are currently present. An agent notifies a site on leaving and a forwarding pointer is left over until agent migration is completed. If agent **B** wants to send a message to **A**, **B** sends the message to all sites in a network. A site **s** discards or forwards the message if **A** is not at **s**.

Query Broadcast: As above but, if agent **B** wants to send a message to **A**, **B** sends a query to all sites in a network asking for the current location of **A**. If site s receives the query and **A** is present at site **s**, then **s** suspends any migration of **A** until **A** receives the message from **B**. A site **s** discards or forwards the query if **A** is not at **s**.

Notification Broadcast: Every site in a network maintains a current guess about agent locations. After migration, an agent distributes in the network, information about its new location. Location information is time stamped. Messages with stale location information are discarded. If site **s** receives a message whose recipient is not at **s** (because it has already migrated or the initial guess was wrong), it waits for information about the agent's new location. Then **s** forwards the message.

Discussion: The cost of communication in Query and Data Broadcasts is high (packets are broadcast in the network) but the cost of migration is low. Query Broadcast saves bandwidth if messages are large or in the case of stream communication. Notification Broadcast has a high cost of migration (the location message is broadcast to all sites) but the communication cost is low and similar to forwarding pointers with pointer chain compaction. In Data and Notification Broadcasts, migration can be fast because there is no synchronization involved (in Query Broadcast migration is synchronized with communication); the drawback is a potential for race conditions if migrations are frequent. Site failures do not disturb the algorithms.

Although they usually assume that the number of sites is too large to broadcast anything, we may allow occasional broadcasts within, e.g. a local Internet domain, or local Ethernet. Broadcasts can be accomplished efficiently in bus-based multiprocessor systems. They are also used in radio networks.

3.2.4 Group Communication:

Algorithm: The agents forming a group maintain a current record about the site of every agent in the group. Agent names form a totally ordered set. We assume communication to take place within a group only. Before migration an agent **A** informs the other agents in the group about its intention and waits for ACKs (containing the number of messages sent to **A**). It then waits for all the messages due to arrive and migrates. After migration it tells the agents it has finished moving. Multicast messages to each agent within a group are delivered in the order in which they are sent (using a first-in-first-out multicast). If **B** wants to send a message to **A**, **B** sends the message to site **s** which is **A**'s current location. During **A**'s migrations (i.e. after sending the ACK to **A**) **B** suspends sending any messages to **A**. If two (or more) agents want to migrate at the same time there is a conflict which can be resolved as follows. Suppose **A** and **C** want to migrate. If **B** receives migration requests from **A** and **C**, it sends ACKs to both of them and suspends sending any messages to agents **A** and **C** (in particular any migration requests). If **A** receives a migration request from **C** after it has sent its own migration request it can either grant ACK to **C** (and **C** can migrate) or postpone the ACK until it has completed moving to a new site. The choice is made possible by ordering agent names.

Discussion: The advantage of this algorithm is that sites can be stateless (the location data are part of agent state). However, in a system with failures the algorithm is more complicated than above. Agents are organized into groups, corresponding to multicast delivery lists, that cooperate to perform a reliable multicast (i.e. if one agent on the delivery list receives a reliable multicast message, every agent on the delivery list receives the message). A precise meaning to the notion of delivery list can be given by using virtual synchrony defined for non-movable groups (Birman et al., 1987). The current list of agents to receive a multicast is called the group view. The group view is consistent among all agents in the group. Processes are added to and deleted from the group via view changes. If agent **A** is removed from the view, the agents remaining in the view would assume that **A** has failed. Virtual synchrony guarantees that no messages from **A** will be delivered in the future (if **A** has not failed it must rejoin the group explicitly under a New name).

A problem is how agents can dynamically join the group, which can change sites. One solution is to leave forwarding pointers, such that agents which want to join (or rejoin) the group can follow them and "catch up" with at least one group member. Another solution is to have one agent within a group (a coordinator or manager) which never migrates. The algorithm for inter-group communication could then use the pointers or coordination agent for delivering messages that cross group boundaries.

The algorithm is suitable for frequent messages (or stream communication) between mobile agents and when migrations are rare. Agent failures and network partitions will not disturb agents which are alive; however, there are detailed subtleties which depend on the semantics of the algorithm implementing virtual synchrony. The group service algorithms for non-movable processes which have been originally proposed, e.g. in ISIS, are costly in terms of control messages and hard to use in networks larger than a LAN. However, they are also examples of scalable group membership and communication services implementing the virtual synchrony semantics, designed for wide-area networks (Keidar et al., 1999).

3.2.5 Hierarchical Location Directory:

Algorithm: A tree-like hierarchy of servers forms a location directory (similar to DNS). Each server in the directory maintains a current guess about the site of some agents. Sites belong to regions, each region corresponds to a sub-tree in the directory (in the extreme cases the sub-tree is simply a leaf-server for the smallest region, or the whole tree for the entire network). The algorithm maintains an invariant that for each agent there is a unique path of forwarding pointers which forms a single branch in the directory; the branch starts from the root and finishes at the server which knows the actual site of the agent (we call this server the "nearest"). Before migration an agent **A** informs the "nearest" server **X1** and waits for ACK. After migration it registers at a new "nearest" server **X2**, tells **X1** it has finished moving and waits for ACK. When it gets the ACK there is already a new path installed in the tree (this may require installing new and purging old pointers within the smallest sub-tree which contains **X1** and **X2**). Messages to agents are forwarded along the tree branches. If **B** wants to send a message to **A**, **B** sends the message to the **B**'s "nearest" server, which forwards it in the directory. If there is no pointer the server will send the message to its parent.

Discussion: Certain optimizations are plausible, e.g. if an agent migrates very often within some sub-tree, only the root of the sub-tree would contain the current location of the agent (the cost of a "move" operation would be cheaper). Moreau (1999) describes an algorithm for routing messages to migrating agents which is also based on distributed directory service. A proposition of Globe uses a hierarchical location service for worldwide distributed objects (Steen et al., 1998). The Hierarchical Location Directory scales better than Forwarding Pointers and Central Servers. Further more, some kinds of fault can be handled more easily (Awerbuch, and Peleg, 1995), and there is also a lightweight crash recovery in the Globe system (Ballintijn et al., 1999).

3.2.6 Arrow Directory:

Some algorithms can be devised for a particular communication pattern. For example, if agents do not require instant messaging, a mail-box infrastructure can be used, where senders send messages to static mailboxes and all agents periodically check mailboxes for incoming messages. Demmer and Herlihy (Demmer and Herlihy, 1998) describe the Arrow Distributed Directory protocol for Distributed shared object systems, which is devised for a particular object migration pattern; it assumes that the whole object is always sent to the object requester. The arrow directory imposes an optimal distributed queue of object requests, with no point of bottleneck. The protocol was motivated by emerging active network technology, in which programmable network switches are used to implement customized protocols, such as application-specific packet routing.

Algorithm: The arrow directory is given by a minimum spanning tree for a network, where the network is modeled as a connected graph. Each vertex models a node (site), and each edge a reliable communication link. A node can send messages directly to its neighbors, and indirectly to non neighbors along a path. The directory tree is initialized so that following arrows (pointers) from any node leads to the node where the object resides.

When a node wants to acquire exclusive access to the object, it sends a message find which is forwarded via arrows and sets its own arrow to itself. When the other node receives the message, it immediately "flips" the arrow to point back to the immediate neighbor who forwarded the message. If the node does not hold the object, it forwards the message. Otherwise, it buffers the message find until it is ready to release the object to the object requester. The node releases the object by sending it directly to the requester, without further interaction with the directory.

If two find messages are issued at about the same time, one will eventually cross the other's path and be "diverted" away from the object, following arrows towards the node (say **v**) where the other find message was issued. Then, the message will be blocked at **v** until the object reaches **v**, is accessed and eventually released.

Chapter 4

Implementation of Mobile Agent Professional (MAP) Communication Tools

4.1 Introduction :

From the previous related work in the mobile agent environment, the most relevant works are these that use the mobile agent as an intermediate node between two computers as a client and a server to do something between them. We use the mobile agent for a more efficient work to establish a communication through local networks to find the <IP, name of device>, and then we can control the device through the mobile agent, which is controlled and managed through built tools which are called Mobile Agent Professional (MAP) tools.

We have, in this thesis, built a small (MAP) system that uses mobile agents, in a local area network (LAN) to:

(1) Find all the active nodes/computers in the network.

The algorithm is as follows: send one (many) mobile agent(s) through the network and search in the routing table about the computers that are registered in it, and return their IP to the server Device in order to be saved in a table list. Saving all nodes results in a document file that contains the <IP, name of device> in the network.

(2) Support communication with devices in the small LAN and management of these devices.

Such a system will enhance the ability to control a network in a simple efficient way.

We then present the steps that we follow to build the (MAP) system. After that, we would talk about the (MAP) system tools to match between each step in the (MAP) system and the tools equivalent to it from the (MAP) system tools.

The First Step is to find all the IP of devices that are near current system, and to find the name of the computer to which the IP belongs. This is achieved as follows: send one (many) mobile agent(s) through the network, and search in the routing table about the computers that are registered in it and return their IP to the server device in order to be saved in a table list. Saving all nodes results in a document file that contains the <IP, name of device> in the network.

The Seconds Step is to find all open ports for all computer near me, in the same network, or out of current networks, using the IP that the mobile agent retrieves from the first step, and using the IP to enter the remote computer without having any privilege in that computer to search for the open port in it and prevent those devices from being attacked by others, and tell it about the open port in its computer by sending services messages to it or upload file that contains the name for all open port using mobile agent as services.

The Third Step is to check the status of all computers which we got their <IP, name of device> in the first step and all open port numbers from the second step to check if the computers are online or offline on the networks to establish the communication between them and save the time which we would spend in sending packet betweens computers if it offline.

The Fourth Step focuses on resolving the IP from the getting computer name or resolve the computer name from the getting IP this step is more efficient that can get the computer name for any computer in or out of current networks and get the details for it .

The Fifth Step focuses on checking the system status for any computers using mobile agent that can send and echo messages and wait for reply from it if it wake up or down that the echo message can have any length and take period of time to return the replay if that times out then the computer is down others it return replay to me.

The Sixth Step is concentrate on sending messages to any computers in the same network or outer it that it related to the first step such that from the IP or computer name we can send service messages to any computer we have IP or computer name without having any privilege on it.

The Seventh step concentrate on concatenating between the first steps and the second steps and third to build tools that can check the IP and Open port and status of the computers in the same times for more efficiency of the system to get all in one but it take more time than others steps as separate functions.

The Eights step is concentrate to get information about class name and Computer ID and Subnet Mask which is related to the first steps, such that from IP we can find class name and others related properties in networks.

Then we would here in this chapter write all the algorithms that we start follow it and using it, to build the system of mobile agent communication, and we would also write each part of system as tools and write an algorithm that clear the idea of those tools as possible.

4.2 (MAP) System Tools Related To IP (Not for Control or Management):

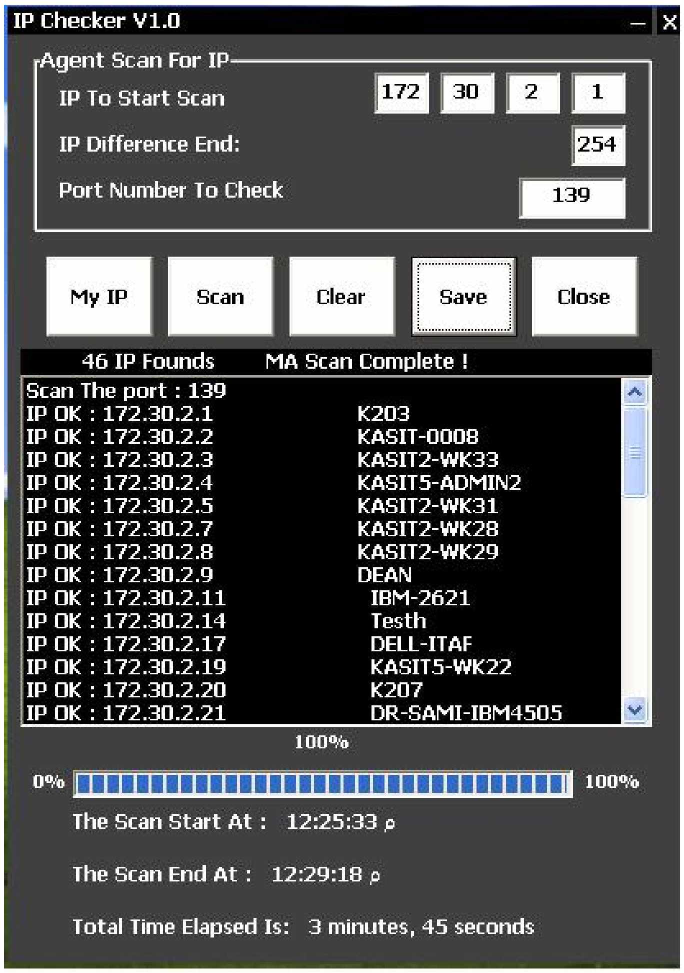
The Mobile agent System that we are developing enables us to retrieve an IP for all computers in the same class, the number of open ports for computers, and the status of the computers. This tool is not used for control or management but for general communication, such as the service of sending messages and retrieving the class name, computer name and online and offline computers.

The small (MAP) system that we are building contains the following tools:

1. Mobile IP Checker (To Get the IP).
2. Mobile Agent Scanner (To Find the Open Ports).
3. Mobile Agent Checker for Online/Offline Computers.
4. Mobile Agent Resolving IP or Computers Name.
5. Mobile Agent Pinger.
6. Mobile Agent Messages.
7. Mobile Agent Scanner Enhanced.
8. Mobile Agent Retrieving Class Name.

4.2.1 Mobile IP Checker (To Get the IP):

These tools help you find all the nodes in the selected networks by selecting the networks which your computer belongs to, or by selecting other networks, taking their class into account, in addition to the number of clients in them. What results from these tools is all the IPs in the selected class of networks, in addition to the name of the computer to which the IP belongs. You can save the result and the number of IPs in a document file, and you will get the starting time and the finishing time that elapsed in searching by the agent. This tool matches the first steps of my thesis. Figure 4.1 illustrates the function of mobile IP checker tools and the result from current networks.



IP To Start Scan 172 30 2 1
IP Difference End: 254
Port Number To Check 139

46 IP Founds MA Scan Complete !

```
Scan The port : 139
IP OK : 172.30.2.1           K203
IP OK : 172.30.2.2           KASIT-0008
IP OK : 172.30.2.3           KASIT2-WK33
IP OK : 172.30.2.4           KASIT5-ADMIN2
IP OK : 172.30.2.5           KASIT2-WK31
IP OK : 172.30.2.7           KASIT2-WK28
IP OK : 172.30.2.8           KASIT2-WK29
IP OK : 172.30.2.9           DEAN
IP OK : 172.30.2.11            IBM-2621
IP OK : 172.30.2.14            Testh
IP OK : 172.30.2.17            DELL-ITAF
IP OK : 172.30.2.19          KASIT5-WK22
IP OK : 172.30.2.20            K207
IP OK : 172.30.2.21            DR-SAMI-IBM4505
```

100%

0% 100%

The Scan Start At : 12:25:33 م

The Scan End At : 12:29:18 م

Total Time Elapsed Is: 3 minutes, 45 seconds

Figure 4.1: Mobile IP checker to get the IP

The algorithm of IP checker helps us build the IP checker tools. This algorithm is one of the algorithms that can find all the IPs around, collecting them from the router table by reading the IP first, checking the class it belongs to, and then starting the search for an IP in the router tables that contains the same IP in a sequential order to avoid losing any IP around, as illustrated in the following steps.

1. Initialization:

```
- Get_My_IP ( )        \\Retrieve The IP of Local Computer
IP ç  W.X.Y.Z          \\Store The IP of Local Computer
```

2. Search for an IP in a Router Table Using the Mobile Agent

Procedure MA_Scan (IP, Portno)   \\ to Scan for IP

```
List_IP as List                    \\ to save founds IP
F as Integer                       \\ to change domain class
Counter_IP as Integer              \\ count number of IP Founds
```

Begin

```
    F=1
    Counter_IP =0
Check_Local_IP (W. X. Y. Z)        \\To check My IP found in which Router Table
    IF Check_Local_IP =True Then
     GoTo L
    Else
```

```
        Go to finish                      \\ current computer is not connect to any networks
  : L
 While Not of End (Router_Table) DO       \\ check Router Table for IP
Check (W.X.Y. F, Portno)                  \\To check IP found in Router Table or not
            IF Check =True Then
            List_IP.additem (W.X.Y. F)    \\add IP found to List
   Retrieve_Name (W.X.Y.F, Portno)        \\retreive The Name for IP that's Founds
   Counter_IP=Counter_IP+1                 \\count number of IP Founds
             If (F <> 254) Then           \\ Check Domain of IP
                  F=F+1
             Else
                   GoTo: Finish
        Else
          Continue
             End If
        Wend
             : Finish
```

End   \\ End of (MA_Scan) Procedure

Procedure Get_My_IP ( )   \\ return My Local IP

```
Show_Status as Boolean        \\ status of Local computer
W.X.Y.Z as String
```

Begin

```
W.X.Y.Z= MY_IP     \\ Store MY IP
Show_Status= Check_Pc_Connected_NT (W.X.Y.Z)    \\ to check if the Local computer is on the
                                                  network or not.
If (Show_Status=True) Then
Read IP from Ethernet TCP/IP Protocol             \\ Get Local IP
Return W. X. Y. Z                                 \\ Return IP If It's Connected To Networks
Else
Return 127.0.0.1                 \\ Local Computer IP That Not Contain Ethernet Card
                                 And Not Connected To the Network.
End If
```

End **\\ End (Get_My_IP) Procedure**

Procedure Check_Pc_Connected_NT (W.X.Y.Z)   \\ check local IP Status

Begin

```
While Not End of (Router_Table) DO               \\ Check IP In router Table or Not.
If (W.X.Y.Z) Found In Router _Table then
 Return True                         \\ the IP Founds
Else
```

```
Return False                              \\ the IP Not Founds
```

End **\\ End of (Check_Pc_Connected_NT) Procedure**

Procedure Check_Local_IP (W.X.Y.Z, Portno) **\\ check IP Founds in Which Router Table**

```
Routers_tables_IP as array of pointers    \\ view all routers tables as array of pointers
                                          for details
```

Begin

```
While not end Of (Router_Table) do        \\ check each router table
Compare_Local (W.X.Y.F, Portno, Routers_Tables_IP)
If Compare_Local=True Then
Return True                               \\ IP founds in one of Router Tables
Else
Return false                              \\ IP Not Founds in any router tables it is offline
GoTo      : Finish
End If
Wend
: Finish
```

End **\\ End of (Check_Local_IP) Procedure**

Procedure Compare_Local (W.X.Y.F, Portno, Routers_Tables_IP) \\ compare IP by routers tables array of Pointers

```
P1 as Pointer

Begin
While not end of (Routers_Tables _IP) do
If (W.X.Y.F=P1.Router_Table_IP) Then
Return True
Else
P1 ç  P1.Next
End If
Wend
If End Of (Routers_Tables_IP) and (W.X.Y.F <> P1.Router_Table_IP) Then
Return false              \\ IP not found in any Router Table
End if

End          \\ End of (Compare_Local) procedure
```

Procedure Check (W.X.Y. F, Portno)   \\ check IP Found in Router Table

```
Router_Table_IP as array                  \\ router table that we Find Local IP in it.

Begin
While not end Of Router_Table do          \\ check router table
Compare (W.X.Y. F, Portno, Router_Table_IP)
```

```
If Compare=True Then
Return True                              \\ IP founds in Router Tables
Else
Return false                    \\ IP Not Founds in router tables it is offline
GoTo    : Finish
End If
Wend
: Finish
```

End \\ End of (Check_Local_IP) Procedure

Procedure Compare (W.X.Y.F, Portno, Router_Table_IP) \\ compare IP by router tables.

```
P1 as Pointer
```

Begin

```
While not end of (Router_Table _IP) do        \\ for router table that we found Current Local IP
                                               in it.
If (W.X.Y.F=P1.Router_Table_IP) Then
Return True
Else
P1 ç  P1.Next
End If
Wend
```

```
If End Of (Router_Table_IP) and (W.X.Y.F <> P1.Router_Table_IP) Then
Return false                    \\ IP not found in Router Table
End if
```

End **\\ End of Compare procedure**

Procedure Retrieve_Name (W. X. Y. F, Portno) **\\ retrieve name of computers in the networks.**

```
P as Pointer
Router_Table_IP as array of pointers
```

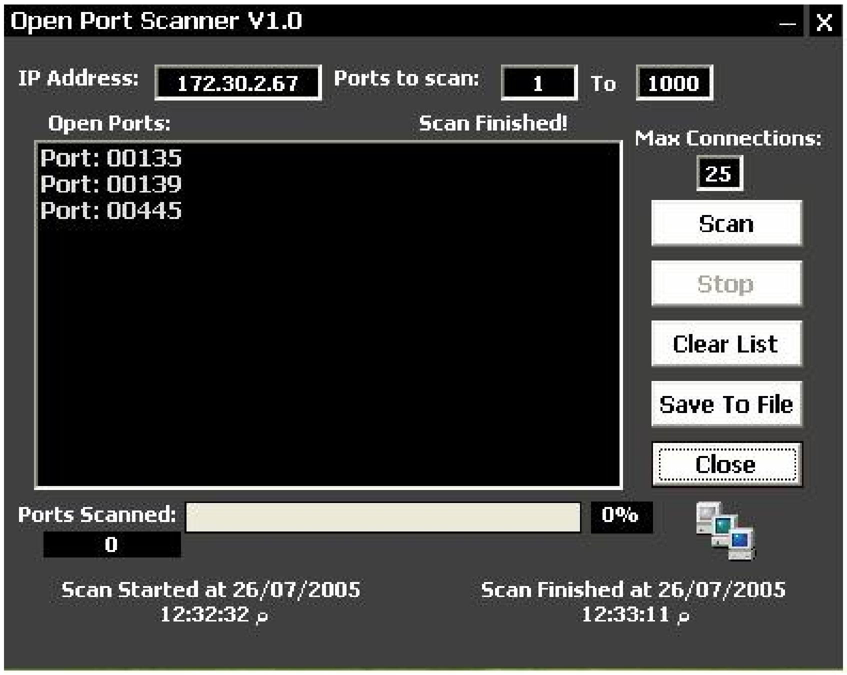
Begin

```
While Not End of (Router_Table) Do
If W.X.Y.F =P.Router_Table_IP Then
Read host name from computer structures in the network
Else
P ç  P.next
End If
Wend
```

End **\\ End of (Retrieve_Name) Procedure**

4.2.2 Mobile Agent Scanner (To Find the Open Ports):

These tools help you find all the open ports in your device directly or by being connected to a remote device or computer using the mobile agent, which searches in the selected computer on the open port by inserting the IP address or the computer name, then sends the number of the open port to the scanner computer (server), which should be first saved on the list. When, the agent finishes searching in the selected computer, you can save the result to the document file to send it as text message at some other time to the selected computer to caution it about the open port which needs to be closed if necessary. The number of ports the agent can scan in a computer ranges from 1- 65535 ports. You can get the starting time and the finishing time that elapsed in searching by the agent, and this tool matches the second step in my thesis. Figure 4.2 illustrates the function of mobile agent scanner tools and the result from remote computers.


Open Port Scanner V1.0

IP Address: 172.30.2.67 Ports to scan: 1 To 1000

Open Ports: Scan Finished!

```
Port: 00135
Port: 00139
Port: 00445
```

Max Connections: 25

Ports Scanned: 0 0%

Scan Started at 26/07/2005 12:32:32 م

Scan Finished at 26/07/2005 12:33:11 م

Figure 4.2: Mobile Agent Scanner to find The Open Ports

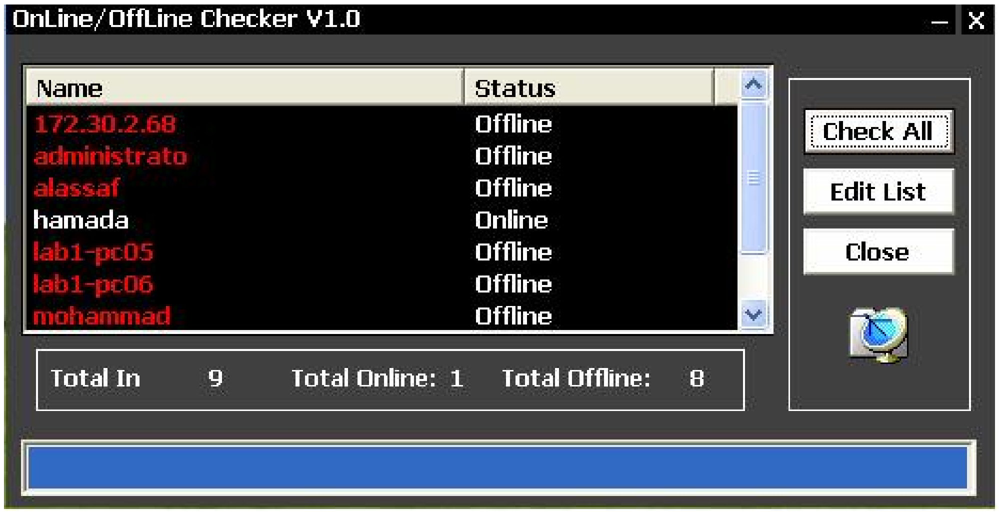
The algorithm of open ports is meant to help us build the open ports tools. This algorithm is one of the algorithms that can find all the open ports for current local IP or remote IPs around through collecting them by checking all ports status on remote computers whether it is close or open by reading the status of each port from the lower port to the upper port. All this is done through the agent that tests each port's status. As illustrated in the following steps.

```
Procedure Open_Port (W.X.Y.Z, Lower, Upper)           \\ check open ports

Status as Boolean
Counter as integer
IP as String

Begin
Counter=0
IP= W.X.Y.Z
For I= lower to upper do
Status= open (IP, I)                 \\ check open port status by send tester agent to test each port
If status =true
Counter =counter+1
Add (I) to the list open port
Else
Continue
End if
End \\ end for I

End   \\ End (open_port) procedure
```

4.2.3 Mobile Agent Checker for Online/Offline Computers:

These tools help you check the status of a group of computers you can select the member of it manually. You must first add the computer that must check the IP and the name of the computer to avoid the duplication between the computer name and the IP for the same computer. For more efficiency, you can add the computer by the IP or name to the list, and then you can check the status by the mobile agent that travels upon the network and checks the router table for the selected computers. If the computer is found in the router table, then the mobile agent returns that the computer is online. Otherwise, it returns that it is offline. The status of computers (whether they are online or offline) on the networks is checked to establish the communications between them and to save the elapsed time to send the packet or not between computers if offline.

This tool matches the third step in my thesis. The status of the computer is showed by the word "offline" or "online" using the red color for the offline line and the white color for the online computers. Figure 4.3 illustrates the function of mobile agent checker tools and the result from remote computers.



Name	Status
172.30.2.68	Offline
administrato	Offline
alassaf	Offline
hamada	Online
lab1-pc05	Offline
lab1-pc06	Offline
mohammad	Offline

Total In 9 Total Online: 1 Total Offline: 8

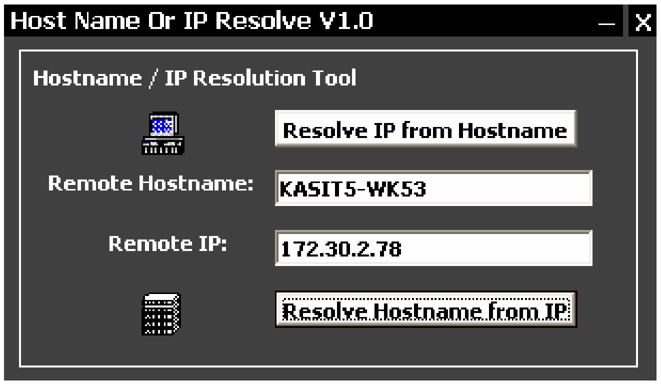
Figure 4.3: Mobile Agent Checker For Online/Offline Computers

The algorithm of online/offline computers helps us build the online/offline computers tools. This algorithm is one of the algorithms that can find all the online/offline status for a list of computers IP around through collecting them by checking all the IPs around by the IP checker. All this is done through the agent, which tests each computer status, as illustrated in the following algorithm:

```
Procedure online_offline (W.X.Y.Z)   \\ check online or offline computers

Status as Boolean
Counter_on, counter_off, portno as integer
IP as String

Begin
Counter_on =0
Counter_off=0
IP= W.X.Y.Z
Check (IP, Portno)                          \\To check IP found in Router Table or not
   IF Check =True Then
   List_IP_online.additem (W.X.Y. F)                  \\add IP online to List
Counter_on = Counter_on +1
Else
List_IP_online.additem (W.X.Y. F)                           \\add IP offline to List
Counter_off = Counter_off +1
End if

End   \\ End of (online_offline) procedure
```

4.2.4 Mobile Agent Resolving the IP or Computer's Name:

These tools help you communicate with other networks or computers in the same network. The function of these tools is to resolve the computer name from the IP for computers in the current network or out of the current network, or vice versa, by returning the IP from the computer name. These tools facilitate getting the name or the IP for one computer at a time, not for all the computers for efficiency and for saving the elapsed time. It matches the fourth step in my thesis. Figure 4.4 illustrates the function of mobile agent resolving the IP or computer's name tools and the result from remote computers.



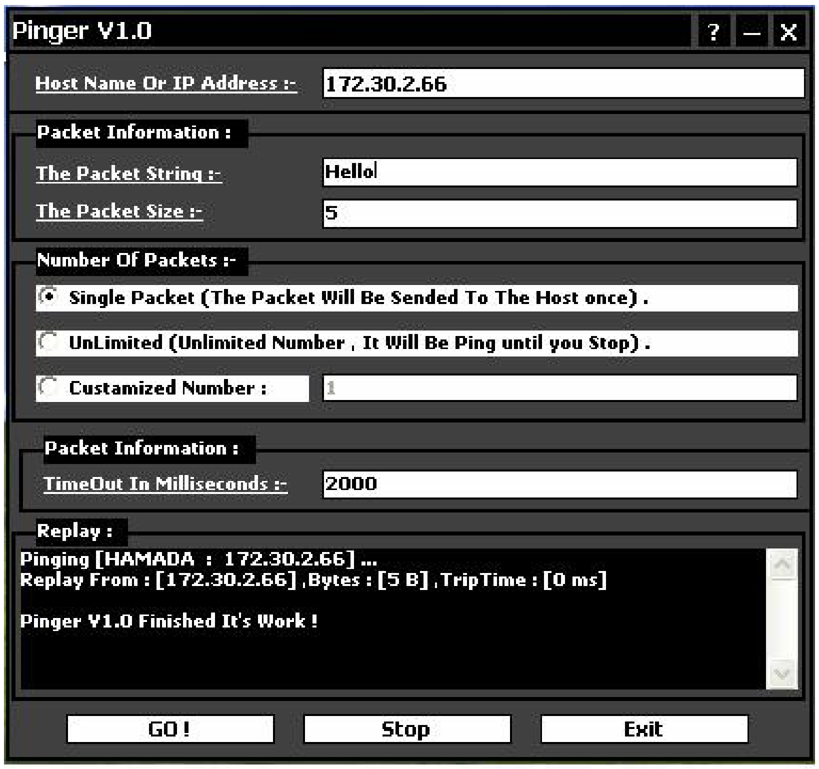
Remote Hostname: KASIT5-WK53

Remote IP: 172.30.2.78

Figure 4.4: Mobile Agent Resolving IP or Computers Name

4.2.5 Mobile Agent Pinger:

These tools checks the system status for any computer using the mobile agent that can send and echo messages to the selected computer by inserting the IP number or the computer name and wait for a reply from it to check whether it is wake up or down, and whether the echo message can have any length and take a period of time to return the reply. If the time is out, then the computer is down. Otherwise, it returns a reply. You can send a single echo message or send the message an infinite number or a custom number of times. This tool matches the fifth step in my thesis. Figure 4.5 illustrates the function of mobile agent pinger tools and the result from remote computers.


Pinger V1.0

Host Name Or IP Address :- 172.30.2.66

Packet Information :

The Packet String :- Hello

The Packet Size :- 5

Number Of Packets :-

[✓ Single Packet (The Packet Will Be Sended To The Host once) .]

UnLimited (Unlimited Number , It Will Be Ping until you Stop) .

Custamized Number : 1

Packet Information :

TimeOut In Milliseconds :- 2000

Replay :

```
Pinging [HAMADA  :  172.30.2.66] ...
Replay From : [172.30.2.66] ,Bytes : [5 B] ,TripTime : [0 ms]

Pinger V1.0 Finished It's Work !
```

Figure 4.5: Mobile Agent Pinger

The algorithm of pinger is meant to help you build the pinger tools. Furthermore, this algorithm helps you check the system status for any computers using the mobile agent, which can send and echo messages to the selected computer by inserting the IP number or the computer name and then waiting for a reply to check whether it is wake up or down. The echo message can have any length and take a period of time to return the reply, as illustrated in the following algorithm:

```
Procedure Pinger (W.X.Y.Z)   \\ check status of the computers

Get replay as Boolean
IP, echo message as String
Number as integer                                   \\ how many numbers of messages send

Begin
IP= W.X.Y.Z
Get replay = Send (IP, echo message, number)         \\To check IP wakeup or down
   IF Get replay =True Then
   List_IP.additem (IP & replay is return from the computer)      \\add IP to List
Else
List_IP.additem (IP & replay is not return from the computer)      \\add IP to List
End if

End   \\ End (pinger) procedure
```

4.2.6 Mobile Agent Messages:

These tools send the message to the selected computer by two ways: by the IP of the computer or by the computer name and the language of the messages (either Arabic or English), which are the languages sent directly by the mobile agent through the servers of messaging in the windows environment. This tool matches the sixth step in my thesis. Figure 4.6 illustrates the function of mobile agent messages tools.

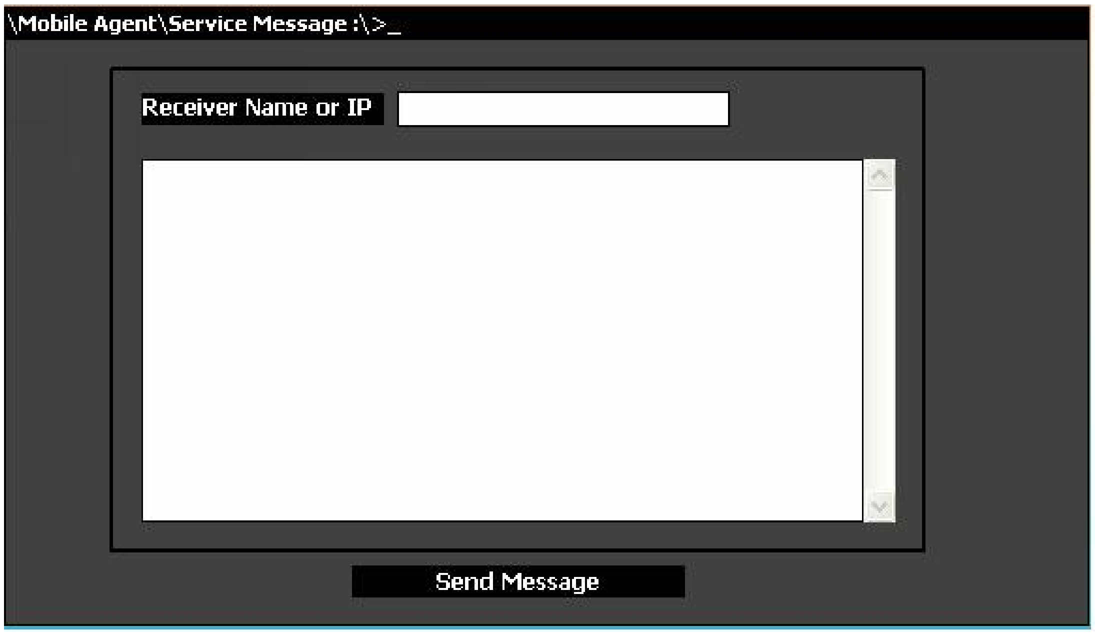


Figure 4.6: Mobile agent Messages

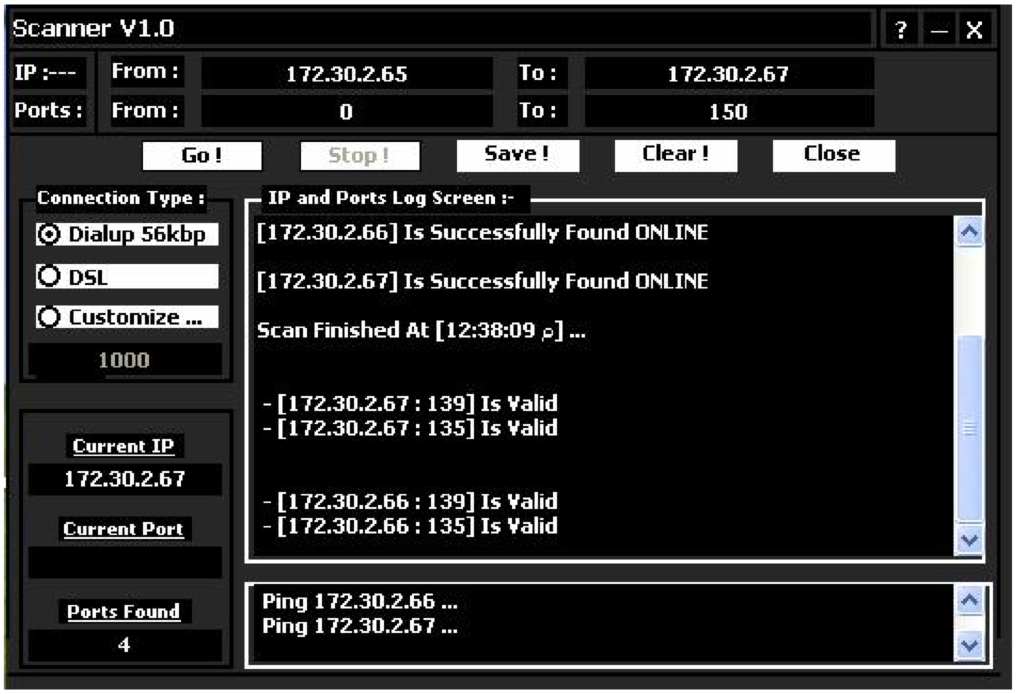
The algorithm of service messages helps you build the service messages tools. Furthermore, it helps you send the message to the selected computer by two ways: by the IP of the computer, or by the computer name. The messenger services must be enabled on a remote computer, and the agent must check the messenger services on that computer and enable them if they were disabled, as illustrated in the following algorithm:

Procedure Service_Messages (W.X.Y.Z) \\ send service message

```
Status as Boolean                 \\ to save result of test the status of message
Message as string                 \\ to store message body

Begin
Status =check_messenger_services        \\ check messenger in remote computer to show the
                                   status of messenger if it enable it is continue other wise
                                  it enabled it first.
If status =true then
send_message (IP, message)        \\ send message directly the messenger is enabled
Else
Enable messenger services and send message again.
End if

End          \\ End of (Service_Messages) Procedure
```

4.2.7 Mobile Agent Scanner Enhanced:

These tools find online/offline computers to check the status and the open port at the same time., They allow you to merge two steps in one step only, needing more time than would be needed with separate tools. The main advantage is to get all online computers and open ports at the same time. Moreover, you can select the connection type for more efficiency for the tools and paying attention to the time needed. This tool matches the seventh step in my thesis. Figure 4.7 illustrates the function of mobile agent scanner enhanced tools and the result from remote computers.



IP :--- From : 172.30.2.65 To : 172.30.2.67
Ports : From : 0 To : 150

(● Dialup 56kbp) ○ DSL ○ Customize ... 1000

```
[172.30.2.66] Is Successfully Found ONLINE

[172.30.2.67] Is Successfully Found ONLINE

Scan Finished At [12:38:09 م] ...


 - [172.30.2.67 : 139] Is Valid
 - [172.30.2.67 : 135] Is Valid


 - [172.30.2.66 : 139] Is Valid
 - [172.30.2.66 : 135] Is Valid
```

Current IP: 172.30.2.67
Ports Found: 4

```
Ping 172.30.2.66 ...
Ping 172.30.2.67 ...
```

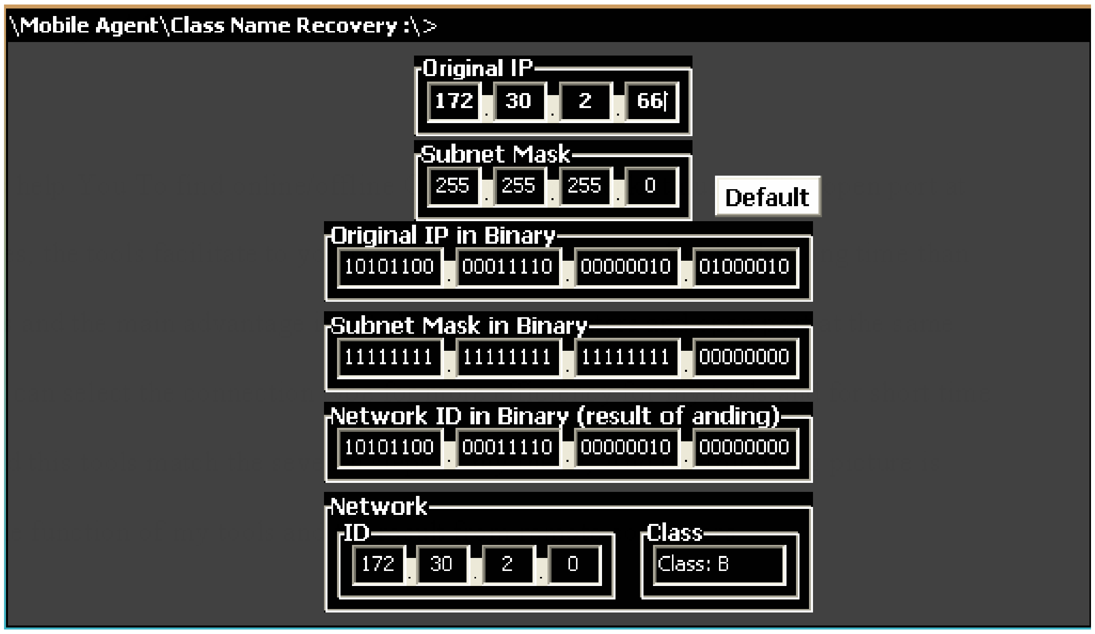
Figure 4.7: Mobile Agent Scanner Enhanced

The algorithm of scanner helps you build the scanner tools and find online/offline computers to check the status and the open ports at the same time, as illustrated in the following algorithm:

```
Procedure Scanner (W.X.Y.Z)   \\ check status of the computers and the
                                   Open port
IP as String
Lower, Upper as integer

Begin
IP= W.X.Y.Z
online_offline (IP)
Open_Port (IP, Lower, Upper)

End          \\ End of Scanner Procedure
```

4.2.8 Mobile Agent Retrieving Class Name:

These tools finds the class name and network id and subnet mask for selected IP then you can check that IP to which class belongs and then you can form a group of IP that have all IP that belongs to the same class for the IP number, these tools match The eights steps which is the final steps for IP and general Information for computers in the same network or different networks. Figure 4.8 illustrates the function of mobile agent retrieving class name tools and the result from remote computers.


\Mobile Agent\Class Name Recovery :\>

Original IP: 172 . 30 . 2 . 66

Subnet Mask: 255 . 255 . 255 . 0

Original IP in Binary: 10101100 . 00011110 . 00000010 . 01000010

Subnet Mask in Binary: 11111111 . 11111111 . 11111111 . 00000000

Network ID in Binary (result of anding): 10101100 . 00011110 . 00000010 . 00000000

Network

ID: 172 . 30 . 2 . 0

Class: Class: B

Figure 4.8: Mobile Agent Retrieving Class Name

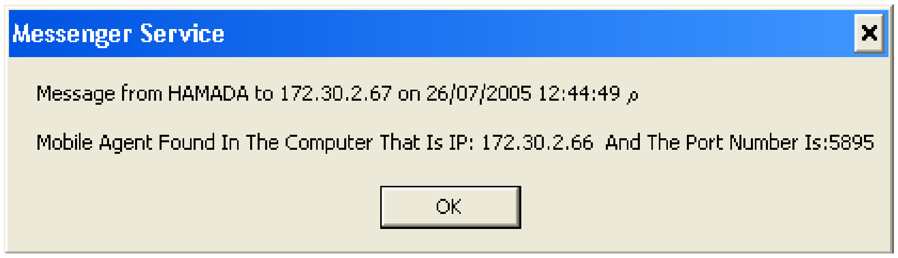
The algorithm of retrieve class name helps us build the retrieve class name tools, this algorithm helps you to find the class name and subnet mask and network id for selected IP then you can check that IP to find the class that belongs to it, to form a group of IP that have all IP that belongs to the same class. As illustrated in the following algorithm.

```
Procedure Retrieve_Class_Name (W.X.Y.Z)   \\ Retrieve Class name for IP

IP_Form as Boolean
Class name string
Network ID and Subnet Mask integer

Begin

IP_Form = check_IP_Form (W.X.Y.Z)        \\ check the form of IP it's Correct Form or Not.
If IP_Form =true then
Return Network ID and Subnet Mask and get the class name that matching IP Domain
Else
Enter correct IP Form
End if

End             \\ End of (Retrieve_Class_Name) Procedure
```

4.3 (MAP) System Tools Related To IP (For Control or Management):

The mobile agent system enables us to have fully control on others computers without having any privilege in that computers the main advantages of my agent is it can send the agent through the email to any person having computers in the same network or in the same class without we know the IP of it's computer or the name of it's computer ,the efficiency of my mobile agent is ,when the person is open my agent (by double click) on it the agent is start it's work, copy it self to other location in the computer and put it self in startup registry folder and check the status of messenger services in the computer and if it disable it enable it directly and start it services and send to me message services contain the IP of that computer and the port number that we can communicate to other computers and with the agent through it, and the agent send infinite message to me if current computer is offline and is stop when current computer become online and at least one message reach to current computer.

The different port number can we send with my agents are for more flexibility of communication and for management more computers at the same time. Figure 4.9 illustrates the message that we receive from the agents in different computers.



Message from HAMADA to 172.30.2.67 on 26/07/2005 12:44:49 م

Mobile Agent Found In The Computer That Is IP: 172.30.2.66 And The Port Number Is:5895

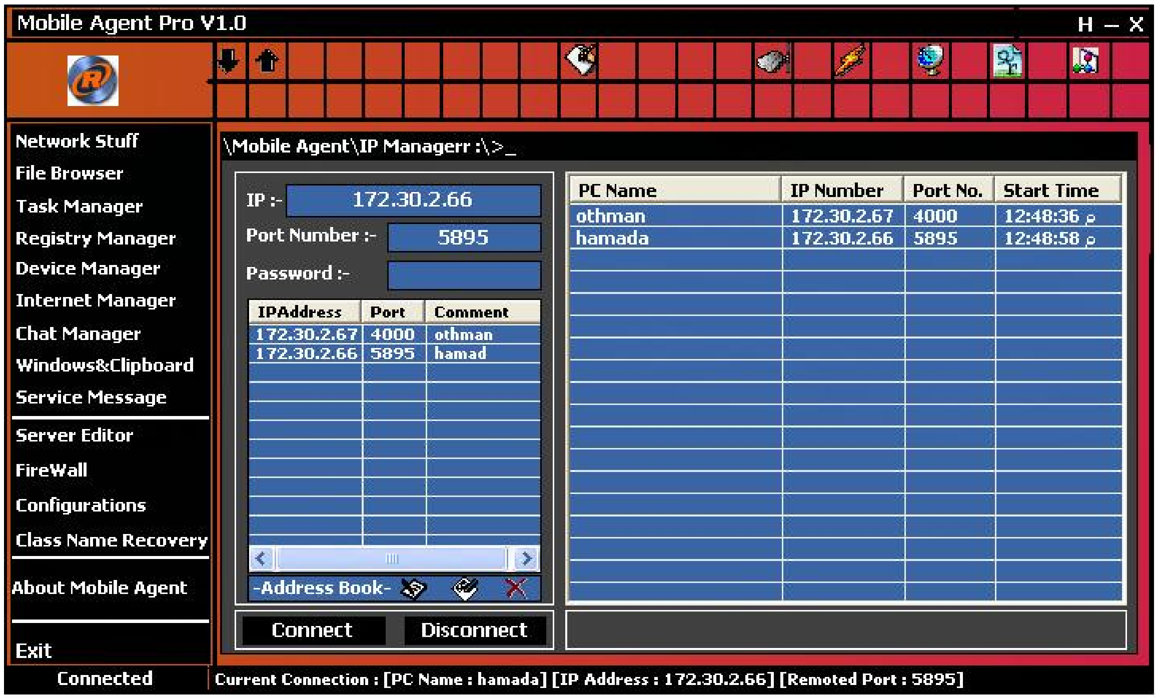
Figure 4.9: Notify Message

The small (MAP) system that we are building contains the following tools:

1. Network Stuff
2. File Browser
3. Task Manager
4. Registry Manager
5. Device Manager
6. Internet Manager
7. Windows Manager
8. Message Box Manager
9. Mobile Agent Editor
10. System Firewall

4.3.1 Network Stuff:

When the agent send to current computer (server) the IP of the computer that is found in it, then we can add the IP that we receive in the message and the port number to connect to the remote computer directly without having any privilege in that computer and with firewall is enabled it that device or disabled we can connect to it .And you can add name to IP to computer to know the agent and port number to which computer is belongs. And you can active any computer in the list by double click in the PC name of the computer in the list, and these tools build from the IP and the port number. Figure 4.10 illustrates the function of network stuff tools.



\Mobile Agent\IP Managerr :\>_

IP :- 172.30.2.66

Port Number :- 5895

Password :-

IPAddress	Port	Comment
172.30.2.67	4000	othman
172.30.2.66	5895	hamad

PC Name	IP Number	Port No.	Start Time
othman	172.30.2.67	4000	12:48:36 م
hamada	172.30.2.66	5895	12:48:58 م

Connected

Current Connection : [PC Name : hamada] [IP Address : 172.30.2.66] [Remoted Port : 5895]

Figure 4.10: Network Stuff

4.3.2 File Browser:

When you connect to the computers that have the mobile agent through the IP number and the port number that we receive through message service, then these function tools is enabled and ready to use .The function of theses tools is to communicate with other computer and navigate all drives in it and do multi types of objectives on it and the types of function and objectives that we can do on the computer is:

- Upload File: this is the main advantage of the tools that we can upload any files that have any size to the computer that we are connected to it to any place in that computer directly by shortest time and easy technique.

- Download File: this is the another main advantage of tools that we can download any files that have any size to current computer directly by short time and easy technique and we can download group of file at the same time and we can stop or pause any file transfer and we can change the priority of the file to get higher or lower priority than other file.

- Copy To/Move To: You can copy or move any file from location to another location in remote computer.

- Execute File: You can run or execute any file or application directly on that computer and change the way that we can run the file by different way.

- Modify Attributes: You can modify the attributes for any file in remote computer such as (read only, hidden, Etc) directly in remote computer.

- Delete File: You can delete any file in remote computer directly by using these tools by shortest time and efficient way.

- Rename File: You can rename any file in remote computer directly by using these tools by shortest time and efficient way.

- New File: You can create any new type of files in remote computer directly by using these tools by shortest time and efficient way.

- Refresh File: Reload file navigations window to see the change if we do any modify of on the files by shortest time and efficient way.

- Search Directory: Search in the remote computer for any types of files by using these tools by shortest time and efficient way.

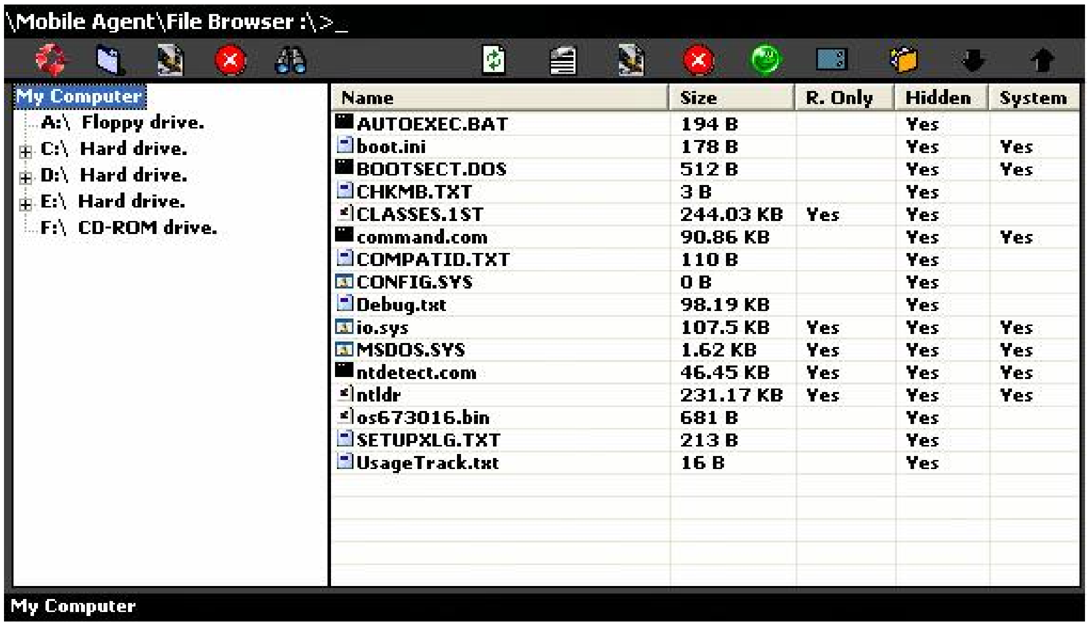
- Remove Directory: Remove any directory and all files that contain it directly in the remote computer by shortest time and efficient way.

- Rename Directory: Change the name of any directory on the remote computer directly by using these tools by shortest time and efficient way.

- New Directory: Create new directory in any location in remote computer by using these tools by shortest time and efficient way.

- Refresh all: Reload all drives information and all files by using these tools by shortest time and efficient way.

When you make double click on any file in any folder you can see text editor that contain information and details about the files. Figure 4.11 illustrates the function of file browser tools and the result that we retrieve from remote computer.



Name	Size	R. Only	Hidden	System
AUTOEXEC.BAT	194 B		Yes	
boot.ini	178 B		Yes	Yes
BOOTSECT.DOS	512 B		Yes	Yes
CHKMB.TXT	3 B		Yes	
CLASSES.1ST	244.03 KB	Yes	Yes	
command.com	90.86 KB		Yes	Yes
COMPATID.TXT	110 B		Yes	
CONFIG.SYS	0 B		Yes	
Debug.txt	98.19 KB		Yes	
io.sys	107.5 KB	Yes	Yes	Yes
MSDOS.SYS	1.62 KB	Yes	Yes	Yes
ntdetect.com	46.45 KB	Yes	Yes	Yes
ntldr	231.17 KB	Yes	Yes	Yes
os673016.bin	681 B		Yes	
SETUPXLG.TXT	213 B		Yes	
UsageTrack.txt	16 B		Yes	

Figure 4.11: File Browser

The algorithm of file browser help us build the file browser tools, the function of this algorithm is to communicate with other computer and navigate all drives in it and do multi types of objectives on it, as illustrated in the following algorithm.

```
Procedure File_Browser (IP, PortNumber)     \\ used to browse other computer

step1, step2, step3 as Boolean
Portnum as integer                 \\ port to scan through
Password as string                 \\ password of the agent

Begin
step1= Open (Portnumber)               \\ To Check the port is Open or not
step2= Check (IP, portnm)              \\ To Check the Computer Online or offline
step3= Check_Pass (Password)           \\ to check the password of the agent

If (step1 and step2 and step 3) = true then
get_drives (IP, PortNumber)            \\ retrieve all drives of remote computer
Else
Goto finish
End if
: finish

End          \\ End of (File_Browser) procedure
```

4.3.3 Task Manager:

These tools are one of the important facilities that the mobile agent facilities it to control and manages all Process in remote computer and this facilities consist of the following part:

- Disable Task Manager: You can disable the task manager (ALT +Ctrl +Del) of the remote computer and we can control all process on it and the person in remote computer can't control any process.
- Enable Task Manager: You can enable the task manager (ALT +Ctrl +Del) of the remote computer and we can control all process on it, and the person in remote computer can control any process.
- Set Priority: You can change the priority for any process in the task manager window in remote computer.
- End Process: you can end any process in the task manager window in remote computer.
- Get Parent: You can get the parent of any process that control and manages the run of application in windows.
- Get Child: You can get the entire child's for the parent of any process that control and manages the run of application in windows.
- Copy to Editor: You can copy all process and application belongs to it to the editor contains all the details about the process and application.

- End Task: You can kill any application running on the remote computer by shortest time and efficient way.

- Maximize Window: You can maximize the window for any application running on the remote computer.

- Minimize Window: You can minimize the window for any application running on the remote computer.

- Restore Window: You can restore the window for any application running on the remote computer.

- Disable Window: You can disable the window for any application running on the remote computer.

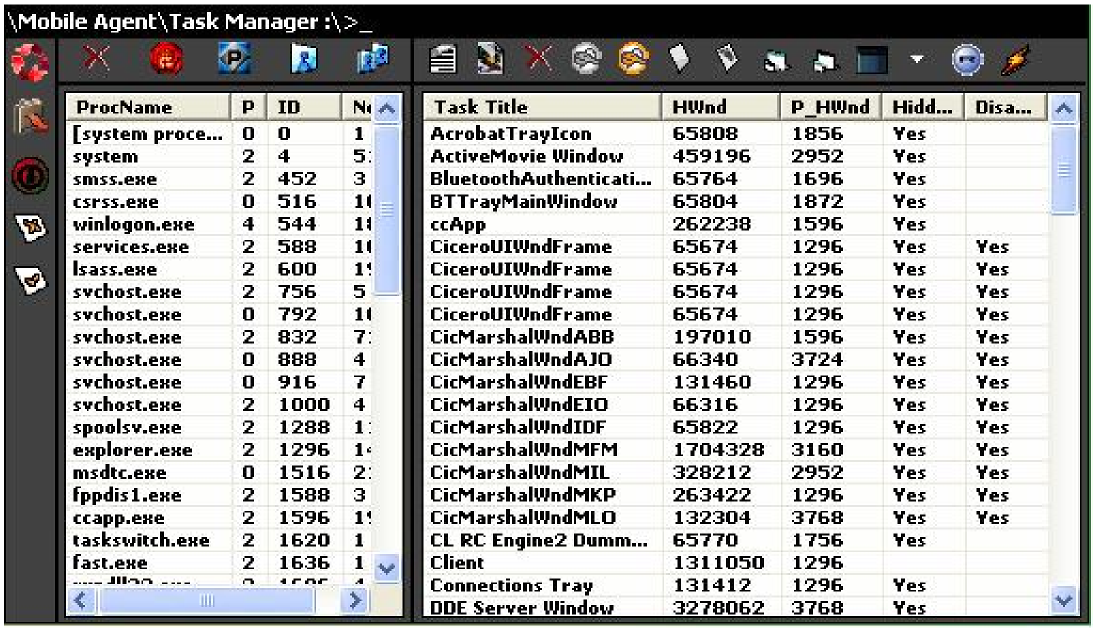
- Enable Window: You can enable the window for any application running on the remote computer.

- Rename Task: You can rename the task of any application running on the remote computers.

- New Task: You can run any application on the remote computers by enter the path of the application by command line.
- Hide Window: You can hide the window for any application running on the remote computer.

- Show Window: You can show the window for any application running on the remote computer.
- Make on Top: You can make the window for any application running on the remote computer on the top comparing to the window of other application.
- Make Not on Top: You can make the window for any application running on the remote computer not on the top comparing to the window of other application.
- Flash Window: You can flash the window of any application.
- End all Process: and all process running on the remote computer and keep my mobile agent running.
- Suspend process: suspend any process on remote computer if you stop or hang up the system.
- Refresh all: reload all process and application from remote computer and retrieve all information about it.

When you make double click on any process or application in the process window then all of the properties and details of the processes or application can see in the text editor that contain information and details about it. Figure 4.12 illustrates the function of task manager tools and the result that we retrieve from remote computer.



\Mobile Agent\Task Manager :\>_

ProcName	P	ID	N
[system proce...	0	0	1
system	2	4	5
smss.exe	2	452	3
csrss.exe	0	516	1
winlogon.exe	4	544	1
services.exe	2	588	1
lsass.exe	2	600	1
svchost.exe	2	756	5
svchost.exe	0	792	1
svchost.exe	2	832	7
svchost.exe	0	888	4
svchost.exe	0	916	7
svchost.exe	2	1000	4
spoolsv.exe	2	1288	1
explorer.exe	2	1296	1
msdtc.exe	0	1516	2
fppdis1.exe	2	1588	3
ccapp.exe	2	1596	1
taskswitch.exe	2	1620	1
fast.exe	2	1636	1

Task Title	HWnd	P_HWnd	Hidd...	Disa...
AcrobatTrayIcon	65808	1856	Yes	
ActiveMovie Window	459196	2952	Yes	
BluetoothAuthenticati...	65764	1696	Yes	
BTTrayMainWindow	65804	1872	Yes	
ccApp	262238	1596	Yes	
CiceroUIWndFrame	65674	1296	Yes	Yes
CiceroUIWndFrame	65674	1296	Yes	Yes
CiceroUIWndFrame	65674	1296	Yes	Yes
CiceroUIWndFrame	65674	1296	Yes	Yes
CicMarshalWndABB	197010	1596	Yes	Yes
CicMarshalWndAJO	66340	3724	Yes	Yes
CicMarshalWndEBF	131460	1296	Yes	Yes
CicMarshalWndEIO	66316	1296	Yes	Yes
CicMarshalWndIDF	65822	1296	Yes	Yes
CicMarshalWndMFM	1704328	3160	Yes	Yes
CicMarshalWndMIL	328212	2952	Yes	Yes
CicMarshalWndMKP	263422	1296	Yes	Yes
CicMarshalWndMLO	132304	3768	Yes	Yes
CL RC Engine2 Dumm...	65770	1756	Yes	
Client	1311050	1296		
Connections Tray	131412	1296	Yes	
DDE Server Window	3278062	3768	Yes	

Figure 4.12: Task Manager

The algorithm of task manager helps us build the task manager tools, this algorithm is one of the important algorithm that the mobile agent facilities it to control and manages all Process in remote computer. As illustrated in the following algorithm.

Procedure Task_Manager (IP, PortNumber)   \\ used to Control Task Manager of the remote computer

```
step1, step2, step3 as Boolean
Portnum as integer                  \\ port to scan through
Password as string                  \\ password of the agent

Begin
step1= Open (Portnumber)                  \\ To Check the port is Open or not
step2= Check (IP, portnm)                 \\ To Check the Computer Online or offline
step3= Check_Pass (Password)              \\ to check the password of the agent
If (step1 and step2 and step 3) = true then
get_Process (IP, PortNumber)                    \\ retrieve all process of remote computer
get_Task (IP, PortNumber, ProcessID)            \\ retrieve all Task of selected Process
Else
Goto finish
End if
: finish

End          \\ End of (Task_Manager) procedure
```

4.3.4 Registry Manager:

This tool enables us to control and manages all registry value in remote computer and this facilities consist of the following part:

- Read Registry in Main Class: Read all the registry key and value in the main class.
- Explore Key: explore selected key in the registry window.
- New Key: Create new key in the registry of main class.
- Delete Tree Key: delete key in main class and all value belongs to it.
- Find Key: search for selected key in main class registry.
- Find value: search for selected value in any key in the main class.
- New Value: Create new value in selected key.
- Rename Value: Rename value in selected key.
- Modify Value: modify value in selected key.
- Delete Value: delete value in selected key.

When you make double click on any key or value in the registry window then all of the properties and details of the key or value can see in the text editor that contain information and details about it. Figure 4.13 illustrates the function of registry manager tools and the result that we retrieve from remote computer.

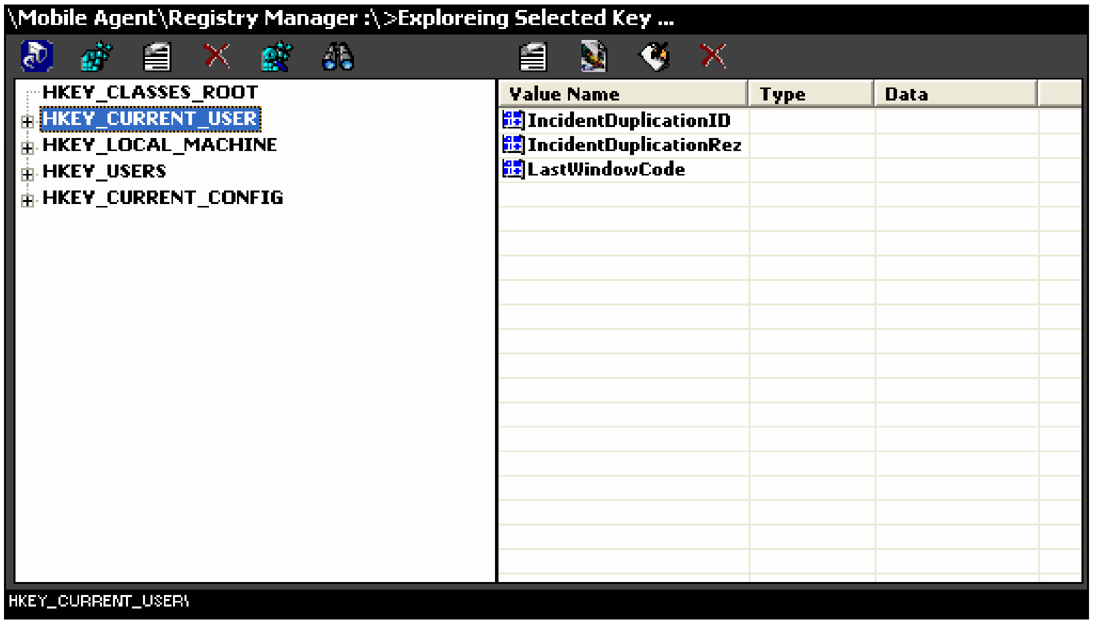


Figure 4.13: Registry Manager

The algorithm of registry manager helps us build the registry manager tools, this algorithm is one of other important algorithm that the mobile agent facilities it to control and manages all registry value in remote computer. As illustrated in the following algorithm.

```
Procedure Registry_Manager (IP, PortNumber)   \\ used to Control Registry
                                                 of the remote computer

step1, step2, step3 as Boolean
Portnum as integer                 \\ port to scan through
Password as string                 \\ password of the agent

Begin
step1= Open (Portnumber)                  \\ To Check the port is Open or not
```

```
step2= Check (IP, portnm)                 \\ To Check the Computer Online or offline
step3= Check_Pass (Password)              \\ to check the password of the agent
If (step1 and step2 and step 3) = true then
get_registry_key (IP, PortNumber)               \\ retrieve all registry key of remote computer
get_registry_value (IP, PortNumber)             \\ retrieve all registry value of remote computer
Else
Goto finish
End if
: finish
```

End **\\ End of (Registry _Manager) procedure**

4.3.5 Device Manager:

These tools allow us to control and manages all device manager in remote computer and this facilities consist of the following part:

- Get Information: Get all information about remote system such as Number of Processors and Processor Type, Processor Level, Processor Version, Processor Speed, Total Physical Memory, Free Physical Memory, Total Memory Page File, Available Memory Page File, Total Virtual Memory, Available Virtual Memory, Country Name, Computer Language, Host Name, Windows Directory, System Directory, Operating System.

- Eject CDROM: Used to eject CD-Room of the remote computer by shortest time and efficient way.

- Close CDROM: Used to close CD-Room of the remote computer by shortest time and efficient way.

- Stop mouse & keyboard: Used to disable mouse & keyboard of the remote computer by shortest time and efficient way.

- Allow mouse & keyboard: Used to enable mouse & keyboard of the remote computer by shortest time and efficient way.

- Turn-off monitor: Used to turn off monitor of the remote computer by shortest time and efficient way.

- Turn-on monitor: Used to turn on monitor of the remote computer by shortest time and efficient way.

- Initialize sound card to be controlled: Used to initialize sound card that will be controlled on remote computer by shortest time and efficient way.

- Take control over sound: Take control of card so that other person can't control it, only we can control on remote computer by shortest time and efficient way.

- Get: Get the master sound volume value on remote computer by shortest time and efficient way.

- Set: Set the volume value of master volume on remote computer by shortest time and efficient way.

- Swap pointer: Swap mouse buttons on remote computer by shortest time and efficient way.

- Return: Return mouse buttons to normal mode on remote computer by shortest time and efficient way.

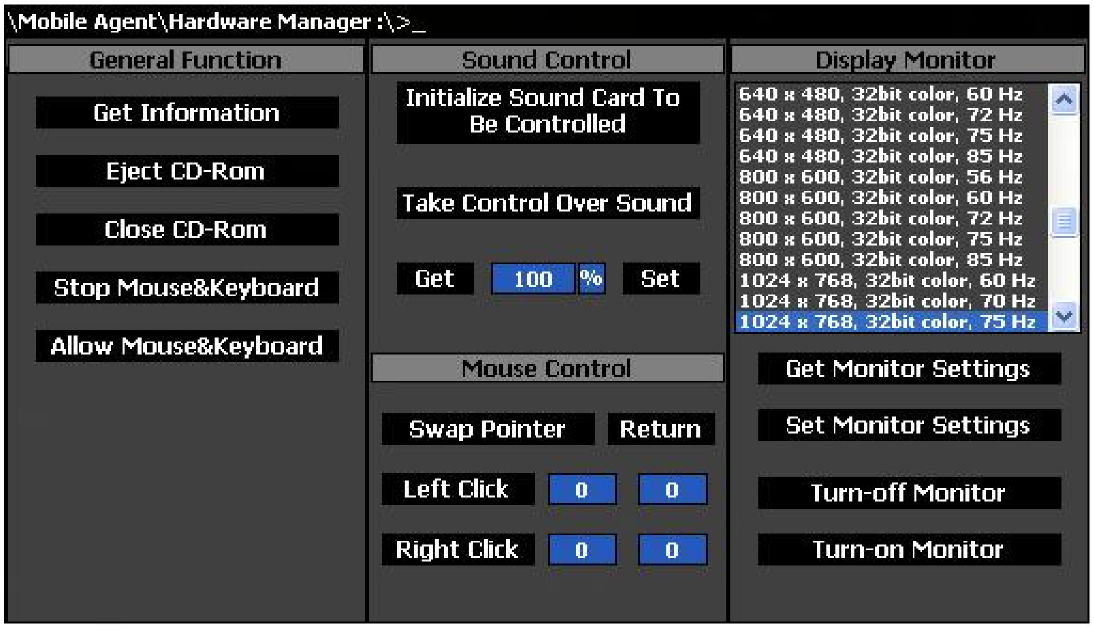
- Left click: Make left click at the specified x, y coordinates on remote computer by shortest time and efficient way.

- Right click: Make right click at the specified x, y coordinates on remote computer by shortest time and efficient way.

- o Get monitor states: Get monitor setting on remote computer by shortest time and efficient way.

- o Set monitor states: Set monitor setting on remote computer by shortest time and efficient way.

Figure 4.14 illustrates the function of device manager tools and the result that we retrieve from remote computer.



```
640 x 480, 32bit color, 60 Hz
640 x 480, 32bit color, 72 Hz
640 x 480, 32bit color, 75 Hz
640 x 480, 32bit color, 85 Hz
800 x 600, 32bit color, 56 Hz
800 x 600, 32bit color, 60 Hz
800 x 600, 32bit color, 72 Hz
800 x 600, 32bit color, 75 Hz
800 x 600, 32bit color, 85 Hz
1024 x 768, 32bit color, 60 Hz
1024 x 768, 32bit color, 70 Hz
1024 x 768, 32bit color, 75 Hz
```

Figure 4.14: Device Manager

The algorithm of device manager helps us build the device manager tools, this algorithm is one of other important algorithm that the mobile agent facilities it to control and manages all devices on remote computer .As illustrated in the following algorithm.

```
Procedure Device_Manager (IP, PortNumber)   \\ used to Control the
                                              Devices on the remote computer.

step1, step2, step3 as Boolean
Portnum as integer                 \\ port to scan through
Password as string                 \\ password of the agent

Begin
step1= Open (Portnumber)                 \\ To Check the port is Open or not
step2= Check (IP, portnm)                \\ To Check the Computer Online or offline
step3= Check_Pass (Password)             \\ to check the password of the agent
If (step1 and step2 and step 3) = true then
Get_information (IP, PortNumber)   \\ get all information from the remote computer
Control_sound (IP, PortNumber, processID)     \\ to control sound device on the remote computer
get_sound (IP, PortNumber, processID)     \\ to get sound level on the remote computer
set_sound (IP, PortNumber, processID)     \\ to set sound level on the remote computer
disabled_mouse_keboard (IP, PortNumber, processID) \\ stop function of mouse and keyboard
enabled_mouse_keboard (IP, PortNumber, processID) \\ enabled functions of mouse and keyboard
get_monitor_setting (IP, PortNumber, processID) \\ get resolution setting from remote computer
set_monitor_setting (IP, PortNumber, processID) \\ change resolution setting for remote computer
off_monitor (IP, PortNumber, processID)     \\ turn-off monitor for remote computer
on_monitor (IP, PortNumber, processID)     \\ turn-on monitor for remote computer
 Swap_mouse (IP, PortNumber, processID)  \\ swap mouse buttons on remote computer
Return_mouse (IP, PortNumber, processID) \\ return mouse buttons on remote computer (normal)
```

```
Left_click (IP, PortNumber, processID, X, Y)        \\ action left click in X, Y coordinate
Right_click (IP, PortNumber, processID, X, Y)       \\ action Right click in X, Y coordinate
Eject_cdrom (IP, PortNumber, processID)    \\ open cd-rom device on remote computer
Close_cdrom (IP, PortNumber, processID)    \\ open cd-rom device on remote computer
Else
Goto finish
End if
: finish
```

End \\ End of (Device _Manager) procedure

4.3.6 Internet Manager:

These tools allow us to control and manages all Internet explorer in remote computer and this facilities consist of the following part:

- Open URL: In the textbox shown write the URL you want to open on the remote computer for example www.yahoo.com by shortest time and efficient way.

- Get browser history: Get the history of the browser or the last opened URLs on remote computer by shortest time and efficient way.

Figure 4.15 illustrates the function of internet manager tools and the result that we retrieve from remote computer.

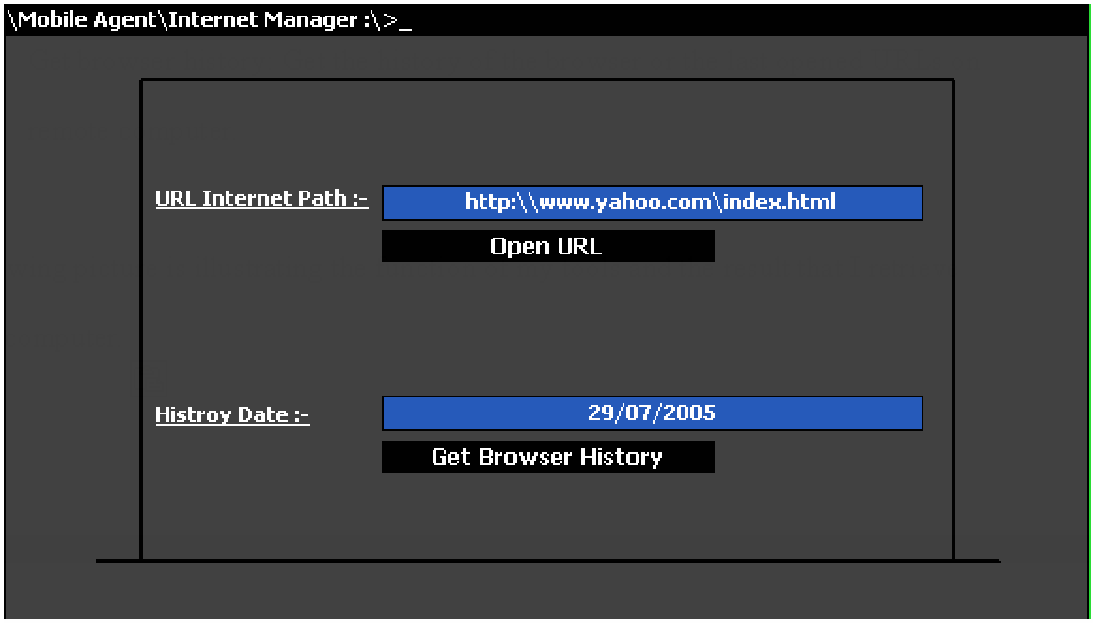


Figure 4.15: Internet Manager

The algorithm of internet manager helps us build the internet manager tools, this algorithm is one of other important algorithm that the mobile agent facilities it to control and manages all internet explorer in remote computer. As illustrated in the following algorithm.

```
Procedure Internet _Manager (IP, PortNumber)   \\ used to Control Internet
                                                   explorer of the remote computer

step1, step2, step3 as Boolean
Portnum as integer                  \\ port to scan through
Password as string                  \\ password of the agent

Begin
step1= Open (Portnumber)                    \\ To Check the port is Open or not
step2= Check (IP, portnm)                   \\ To Check the Computer Online or offline
step3= Check_Pass (Password)                \\ to check the password of the agent
If (step1 and step2 and step 3) = true then
Open_ Internet _URL (IP, PortNumber)        \\ open Internet URL in remote computer
Get_ Internet _ history (IP, PortNumber)     \\ retrieve all Internet history of remote computer
 Else
Goto finish
End if
: finish

End          \\ End of (Internet _Manager) procedure
```

4.3.7 Windows Manager:

These tools allow us to control and manages windows and clipboard in remote computer and these facilities consist of the following part:

- Logoff: helps you to logoff the remote computer from your local machine to through the mobile agent.
- Restart: That helps you to restart the remote computer from your local machine through the mobile agent.
- Shutdown: That help you to shutdown the remote computer from your local machine through the mobile agent to shutdown it.
- Set Time: Change The Time of the remote computer through the mobile agent directly by shortest time and efficient way.
- Set Date: Change the Date of the remote computer through the mobile agent directly by shortest time and efficient way.
- Get Clipboard: get the content of the clipboard of the remote computer by shortest time and efficient way.
- Set Clipboard: Change the content of the clipboard of the remote computer by shortest time and efficient way.

- Clear Clipboard: Clear the content of the clipboard of the remote computer by shortest time and efficient way.

- Disable Clipboard: Disable the clipboard of the remote computer by shortest time and efficient way.

- Enable Clipboard: Enable the clipboard of the remote computer by shortest time and efficient way.

Figure 4.16 illustrates the function of windows manager tools and the result that we retrieve from remote computer.

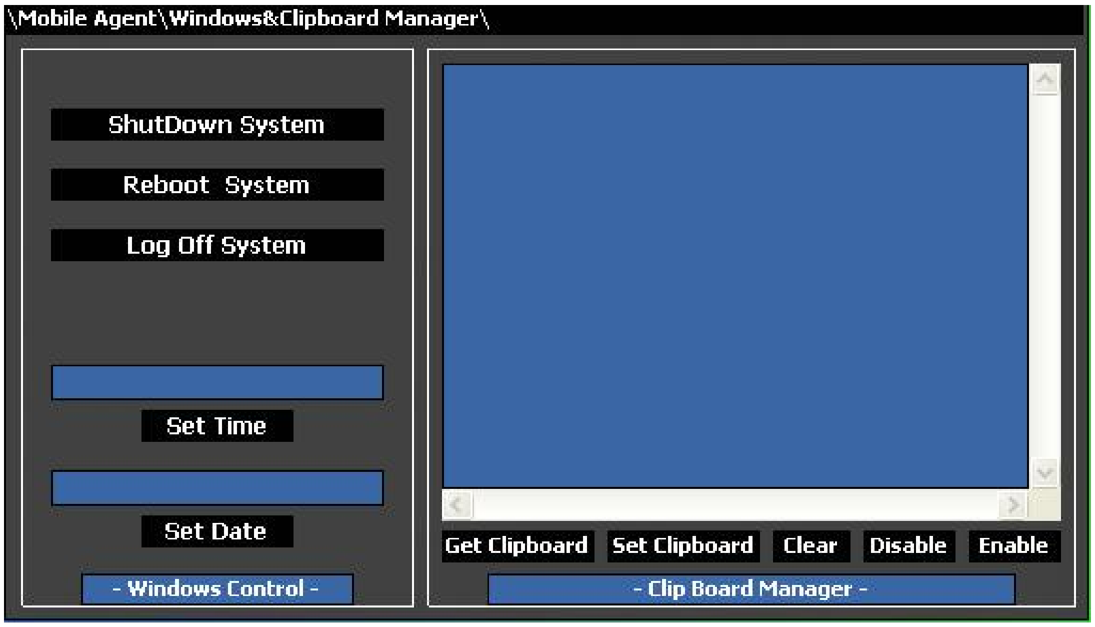


Figure 4.16: Windows Manager

The algorithm of windows manager helps us build the windows manager tools, this algorithm is one of other important algorithm that the mobile agent facilities it to control and manages some windows properties on remote computer. As illustrated in the following algorithm.

Procedure Windows _Manager (IP, PortNumber) \\ used to Control the Windows on the remote computer

```
step1, step2, step3 as Boolean
Portnum as integer                          \\ port to scan through
Password as string                          \\ password of the agent


Begin
step1= Open (Portnumber)                    \\ To Check the port is Open or not
step2= Check (IP, portnm)                   \\ To Check the Computer Online or offline
step3= Check_Pass (Password)                \\ to check the password of the agent
If (step1 and step2 and step 3) = true then
Logoff (IP, PortNumber, processID)  \\ logoff the remote computer
Restart (IP, PortNumber, processID)     \\ restart the remote computer
Shutdown (IP, PortNumber, processID)        \\ shutdown the remote computer
get_clipboard (IP, PortNumber)              \\ get contents of clipboard on remote computer
set_clipboard (IP, PortNumber)              \\ set contents of clipboard on remote computer
Enabled_ clipboard (IP, PortNumber)         \\ enabled clipboard on remote computer
disabled_clipboard (IP, PortNumber)         \\ disabled clipboard on remote computer
clear_clipboard (IP, PortNumber)            \\ clear contents of clipboard on remote computer
```

```
set_Time (IP, PortNumber)          \\ change the time of remote computer
set_Date (IP, PortNumber)          \\ change the date of remote computer
 Else
Goto finish
End if
: finish
```

End   \\ End of (Windows _Manager) procedure

4.3.8 Message Box Manager:

These tools allow us to control and manages windows and event message that can send to the remote computer directly through the mobile agent the title and the body of the message can be in Arabic or English language and it have any length and the type of the message is different such as information message or critical messages or questions messages and other types and answer button is different form such as ok and cancel and ignore an other type you can test the message before send it and the action of the user that do for the message retrieve to me from the remote computer. Figure 4.17 illustrates the function of message box manager tools and the result that we retrieve from remote computer.

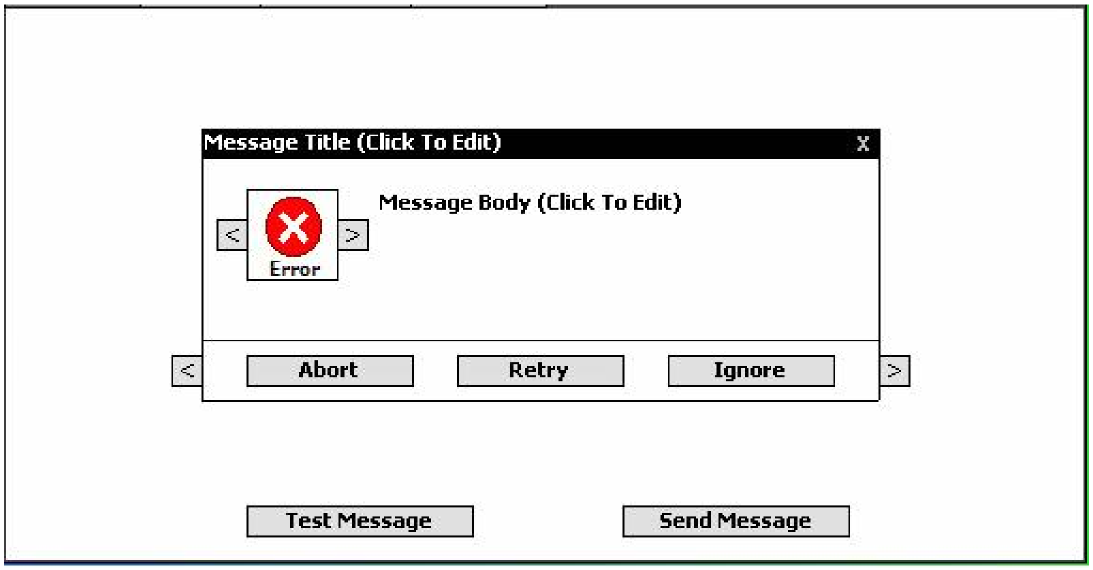


Figure 4.17: Message Box Manager

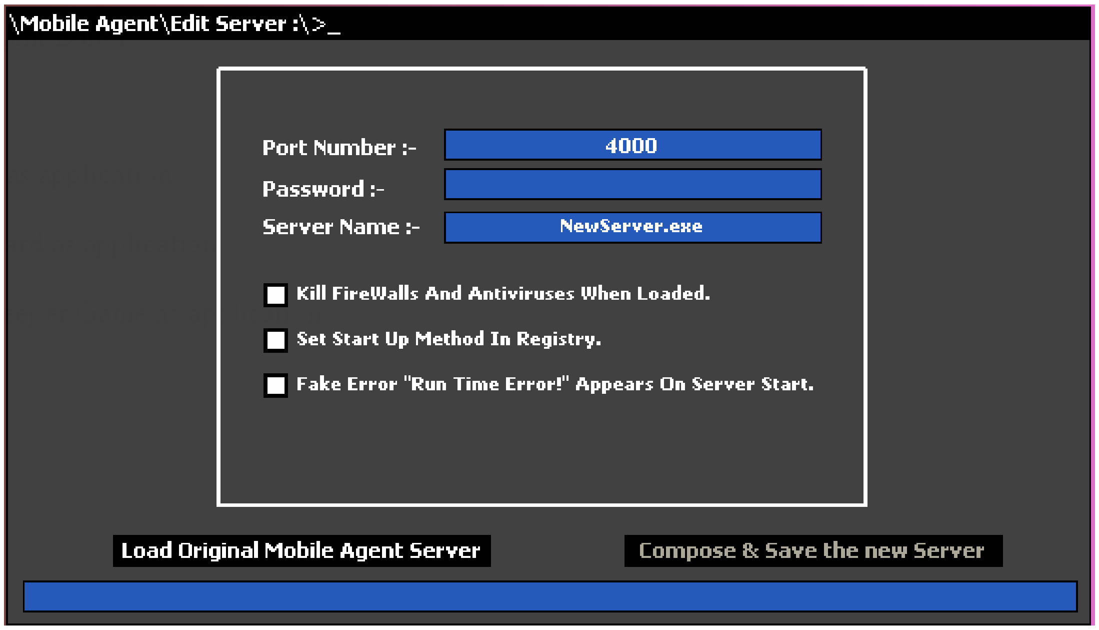
The algorithm of message box manager helps us build the message box manager tools, this algorithm is one of other important algorithm that the mobile agent facilities it to control and manages event message that can send to the remote computer directly through the mobile agent, as illustrated in the following algorithm.

```
Procedure Message_Box_Manager (IP, PortNumber)   \\ used to Control the
                         message box that send to the remote computer

step1, step2, step3 as Boolean
Title, body as string
Picture_ID, Type_ID as integer      \\ type of picture and button that appear to the user
Portnum as integer                  \\ port to scan through
Password as string                  \\ password of the agent

Begin
step1= Open (Portnumber)            \\ To Check the port is Open or not
step2= Check (IP, portnm)           \\ To Check the Computer Online or offline
step3= Check_Pass (Password)        \\ to check the password of the agent.
If (step1 and step2 and step 3) = true then
Test_message (Title, Body, Picture_ID, Type_ID)     \\ test message before send to the remote
                                                        computer.
Send_message (IP, PortNumber, processID, Title, Body, Picture_ID, Type_ID) \\ send message
                                                        to the remote computer.
Get_action_message (IP, PortNumber, messageID)  \\ Return the action of any button that user
                                                        click when receive message.
```

```
Else
Goto finish
End if
: finish

End            \\ End of (Message_Box_Manager) procedure
```

4.3.9 Mobile Agent Editor:

Mobile agent editor is one of the tools that must be create and it's done the main problem of the mobile agent is the fixed port number that the mobile agent can be control through it and when you send multi mobile agent to different computers you must know which mobile agent send to which computer then by this tools you can change and create different port number for the mobile agent that each computer can send to it mobile agent having port number different to the port number for other mobile agent and you can set of others properties to the mobile agent such as adding password to mobile agent and The importance of adding a password for a mobile agent is to help you to prevent any other one from using that mobile agent and other properties is set start up method in registry to run agent at start up and kill firewalls and antivirus when loaded if you want but we not like this and Fake error or run time error appear when server start if you want system message error when agent run .and you must load the original mobile agent that you will find it in the program folder and add other properties to it and then save the new mobile agent again. And the following Figure is illustrating the function of mobile agent editor tools and the result that we retrieve from remote computer.


\Mobile Agent\Edit Server :\>_

Port Number :- 4000

Password :-

Server Name :- NewServer.exe

☐ Kill FireWalls And Antiviruses When Loaded.

☐ Set Start Up Method In Registry.

☐ Fake Error "Run Time Error!" Appears On Server Start.

Load Original Mobile Agent Server
Compose & Save the new Server

Figure 4.18: Mobile Agent Editor

The algorithm of mobile agent editor helps us build the mobile agent editor tools, this algorithm is one of important algorithm that we are built and that can solve the problem of fixed port number of mobile agent and many other tools that is still problem in computer networks filed as security of the agent and we are solved by mobile agent editor, as illustrated in the following algorithm

```
Procedure Agent_Editor (agent_name, PortNumber, Password)   \\ used to
                                          Control the mobile agent configuration.

Kill, startup, fake_error as integer          \\ value of selected options in main editor
Portnum as integer                            \\ port to scan through
Password as string                            \\ password of the agent
Name as string                                \\ the name of new agent

Begin
load_original_agent ( )              \\ load original version of mobile agent to edit content.
check_kill_anti_firewall ( )         \\ check the option of kill antivirus and firewall
check_start_up_registry ( )          \\ check the option of start up method in registry
check_fake_error ( )                 \\ check fake error message.
compose_agent (Portnum, Password, Name, Kill, startup, fake_error)      \\ create the new  mobile
                                                                  agent configuration.

End          \\ End of (Agent_Editor) procedure
```

4.3.10 System Firewall:

The Firewall aims to save current computer from outside hacking and for control the mobile agent communication and allowed to it to communicate with current computer and we can show if other application wanted to communicate with me and allowed it or block it as we like and it is for more efficiency and security for my mobile agent such as add password in the mobile agent editor. These tools can run from configuration and enable firewall engine if you like if you enable it the following part of firewall is seeing:

- File Name: the file name of the firewall found.
- Type : TCP or UDP
- State : listening or closed
- Access :
    - Ask for: for applications you will ask about their connection every time.
    - Allowed: programs that are always allowed to access the network.
    - Denied: programs that are always prevented from accessing network.
- TCP: Presents applications that are connected to TCP ports
- UDP: Presents all applications that are connected to UDP ports
- System files: Presents system files.
- Listening: Presents all listening applications.
- Block: Choose from the list the program that you will always prevent its network access.
- End process: End selected connection's work.
- Suspend: For applications that can't be blocked you can suspend the connection for some time.
- Resume: Resumes the suspended connections work

- Block All: Press block all if you want to prevent all the programs from accessing the network.
- Delete File: delete file from the allowed or block list.
- (?) File Details: Presents the details of the selected file such as Full path and Size and Type and Creation time.
- Add: Click adds to add new applications that you want to allow, block or ask for.
- Allow: Choose the program from the list that you will allow to accessing the network.
- Ask For: Choose from the list the program that you always want to be asked about its connection.
- Block: Choose from the list the program that you will always prevent its network access.

Figure 4.19 illustrates the function of system firewall tools and the result that we retrieve from remote computer.

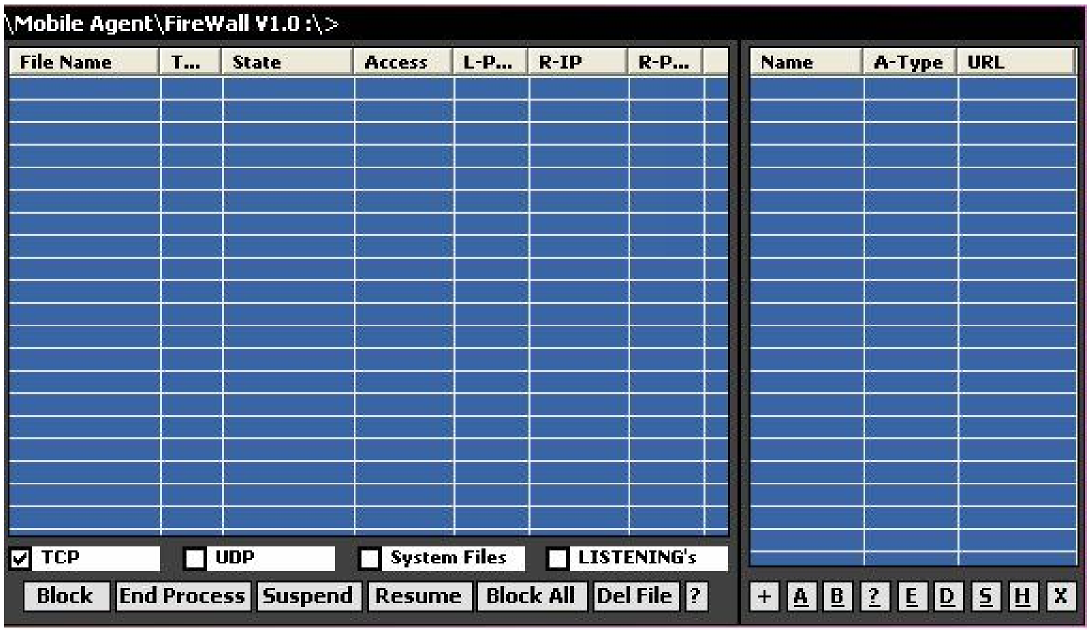


Figure 4.19: System Firewall

The algorithm of system firewall helps us build the system firewall tools, this algorithm helps us save your computer from outside hacking and for control the mobile agent communication and allowed to it to communicate with current computer and we can show if other application wanted to communicate with me and allowed it or block it. As illustrated in the following algorithm

Procedure System_Firewall (IP, PortNumber) \\ used to Control the system firewall.

```
step1, step2, step3, step4 as Boolean
Portnum as integer                  \\ port to scan through
Password as string                  \\ password of the agent

Begin
step1= Open (Portnumber)            \\ To Check the port is Open or not for mobile agent
step2= Check (IP, portnm)           \\ To Check the Computer Online or offline for
                                    mobile agent to allowed access of it.
step3= Check_Pass (Password)        \\ to check the password of the agent.
If (step1 and step2 and step 3) = true then
Step4 = Test_status_Firewall (IP_Local)     \\ check firewall status.
If (step4) = true then
Test_system (IP_local, portnumber_local)    \\ test system of local computer.
Take_action (IP_local, portnumber_local, block, allowed) \\ take action when process want to
                                    communicate with current local computer.
Listen (TCP, UDP, System_files)     \\ listen for communication in the TCP or UDP or System
                        files to manage them
Else                    \\ for check firewall status
Enabled firewall then rechecks system again.
Else
Goto finish
```

```
End if
: finish

End            \\ End of (System_Firewall) procedure
```

4.4 (MAP) System Tools Related To IP (As Application):

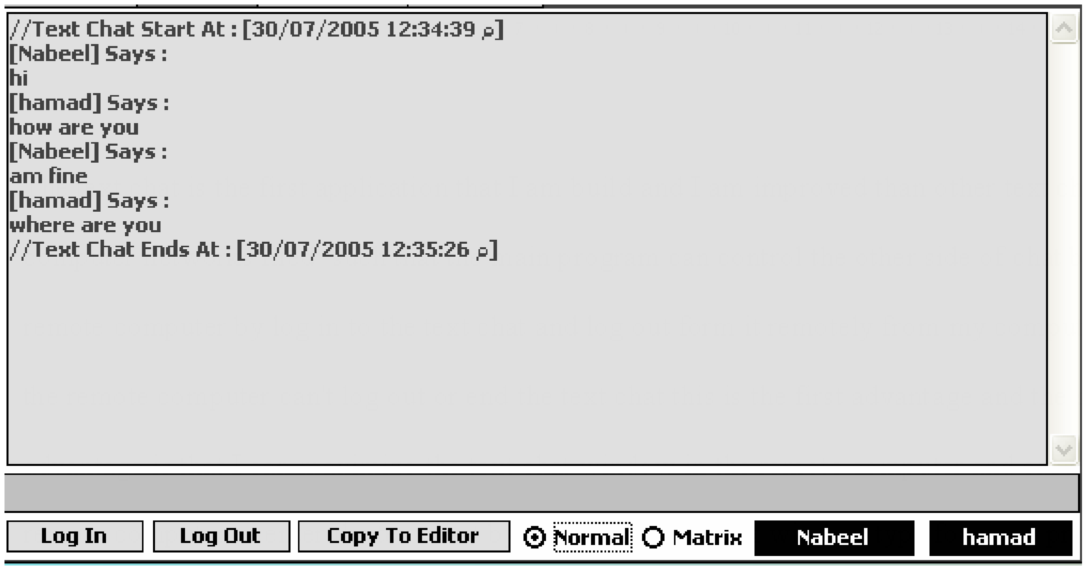
The mobile agent system help us to have full control on others computers without having any privilege in that computers this is the main advantage of my agent then from this point we are built an three applications that correspond to IP and improve it using the mobile agent to reach to the powerful management and communication between computers in the same network

The small (MAP) applications system that we are building contains the following tools application which is very improved than other application in computer networks:

1. Text Chat as application
2. White board as application
3. Mine Sweeper Game as application

4.4.1 Text Chat:

The text chat is an improvement over other text chat in computer networks filed such that the main program can control the other side of chat box in remote computer by log in to the text chat and log out from it remotely from current computer and the remote computer can't log out or end the text chat this is the first advantage and the second advantage are that we can maximize the text chat window in the remote computer and enforce the remote computer to enter the text chat by selecting the matrix window type from my program and the third are we can enter any name for current computer and remote computer to enter the chat room and when the other person on remote computer write to me message through the chat manager we hear sound to notify me that is send message to me and we receive new message from him and the forth we can copy all the text chat editor contents to my editor and then save it by any extensions. Figure 4.20 illustrates the function of text chat tools and the result that we retrieve from remote computer.



```
//Text Chat Start At : [30/07/2005 12:34:39 م]
[Nabeel] Says :
hi
[hamad] Says :
how are you
[Nabeel] Says :
am fine
[hamad] Says :
where are you
//Text Chat Ends At : [30/07/2005 12:35:26 م]
```

Figure 4.20: Text Chat

The algorithm of text chat helps us build the chat manager tools, this algorithm is the first algorithm that we are built improved than other algorithm of text chat in computer networks filed such that the main program can control the other side of chat box in remote computer by log in to the text chat and log out from it remotely from current computer and the remote computer can't log out or end the text chat, as illustrated in the following algorithm.

```
Procedure Chat _Manager (IP, PortNumber)   \\ used to Control the
                                              Chat on the remote computer.

step1, step2, step3 as Boolean
ChatID as integer                 \\ type of the chat window
Portnum as integer                      \\ port to scan through
Password as string                      \\ password of the agent

Begin

step1= Open (Portnumber)                \\ To Check the port is Open or not
step2= Check (IP, portnm)               \\ To Check the Computer Online or offline
step3= Check_Pass (Password)            \\ to check the password of the agent
If (step1 and step2 and step 3) = true then
Start_chat (IP, PortNumber, chatID)  \\ start chat window on remote computer
End_chat (IP, PortNumber, chatID)   \\ end chat window on remote computer
Else
```

```
Goto finish
End if
: finish
```

End **\\ End of (Windows _Manager) procedure**

4.4.2 White Board:

The White Board is better than any other white board in computer networks filed such that the main program can control the other side of White Board box in remote computer by log in to the White Board and log out from it remotely from current computer and the remote computer can't log out or end the white board this is the first advantage and the second advantage is that we can draw not by using pen alone but by using spray tools and line tools and we can change the size of the pen or spray tools or the line tools and we can change the color of the pen or spray tools or line tools and we can make new page and we can the image that we draw with the remote computer in any location in current computer. Figure 4.21 illustrates the function of white board tools and the result that we retrieve from remote computer.

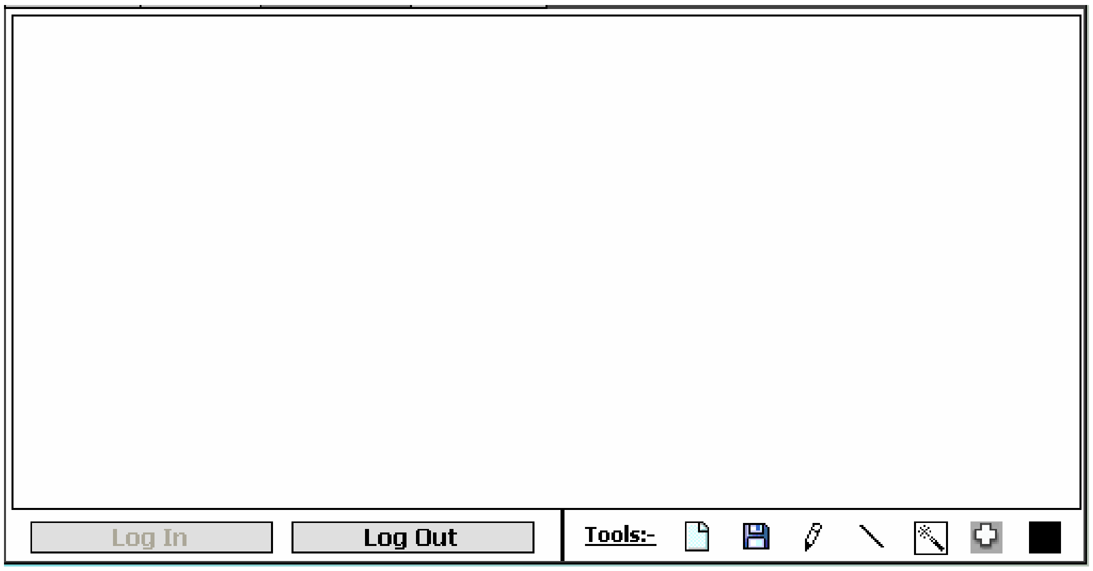


Figure 4.21: White Board

The algorithm of white board helps us build the white board manager tools, this algorithm is the second algorithm that we are built and improved than other white board algorithm in computer networks filed such that the main program can control the other side of white board box in remote computer by log in to the white board and log out from it remotely from current computer and the remote computer can't log out or end the white board, as illustrated in the following algorithm.

```
Procedure White_Board_Manager (IP, PortNumber)   \\ used to Control the
                                                 white board on the remote computer

step1, step2, step3 as Boolean
Board_tools_ID as integer          \\ tools to use in drawing on white board
Portnum as integer                 \\ port to scan through
Password as string                 \\ password of the agent
Begin
step1= Open (Portnumber)               \\ To Check the port is Open or not
step2= Check (IP, portnm)              \\ To Check the Computer Online or offline
step3= Check_Pass (Password)           \\ to check the password of the agent
If (step1 and step2 and step 3) = true then
Start_white_board (IP, PortNumber, board_tools_ID)     \\ start white board window on remote
                                                       computer.
End__white_board (IP, PortNumber, board_tools_ID)      \\ end white board window on remote
                                                       computer.
Else
Goto finish
End if
: finish
End          \\ End of (White_Board_Manager) procedure
```

4.4.3 Mine Sweeper:

The mine sweeper is an improvement over other mine sweeper in computer networks filed such that Mine sweeper is a game built by Microsoft Corporation under Windows area. This game's idea revolves around avoiding the mine, and it can be played by one person only on the same computer. The idea in the game we represent is different in that two persons can play the game on two computers being connected to the network through the mobile agent, which would facilitate the communication between the two persons and make the game more controlled. The game contains 45 mines. If one person finds 23 mines, this would mean that he has won and others have lost the game. This game idea and controller through the mobile agent is one of the powerful advantage applications to communicate two computers through the mobile agent. The main program can control the other side of Mine sweeper box in remote computer by log in to the Mine sweeper game and log out from it remotely from current computer and the remote computer can't log out or end the Mine sweeper this is one of the advantage. and when the other person on remote computer play and put selection for the mine we hear sound to notify me he is play .and when we Find or other person find Mine he still play otherwise the game role is exchange to other person or to me .And the two boxes show the grade of me and others person to tell me and the others person which one his grade is grater than others to win the game and the others lost the game and game over .Figure 4.22 illustrates the function of mine sweeper tools and the result that we retrieve from remote computer to show the efficiency of my game using the mobile agent as intermediate node to establish and control communication between current computer and other computer in the networks .

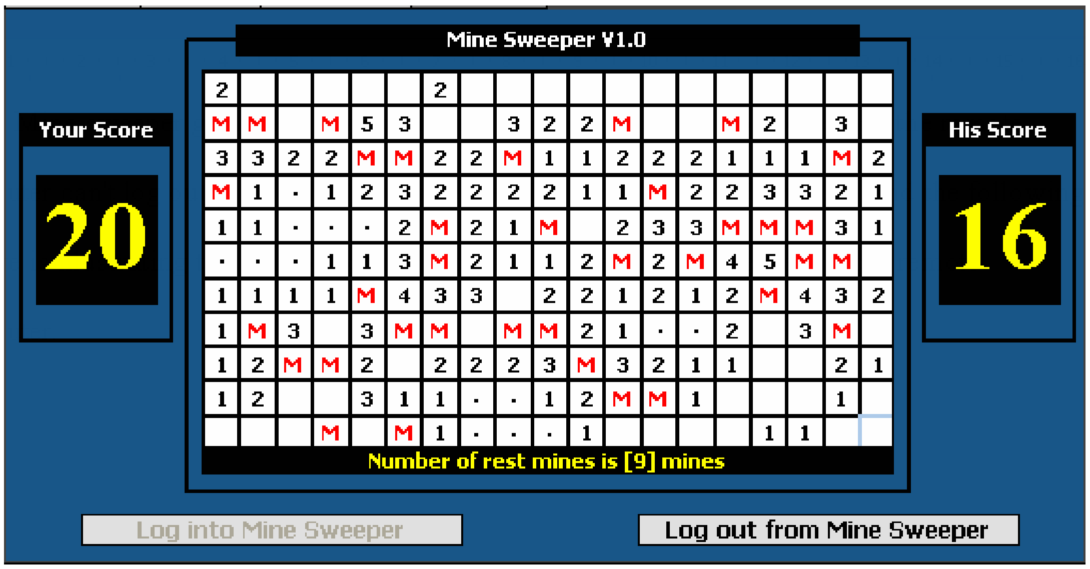


Figure 4.22: Mine Sweeper

The algorithm of mine sweeper helps us build the mine sweeper manager tools, this is the third algorithm that we are built and improved than other Mine sweeper algorithm in computer networks filed such that mine sweeper is a game is build by Microsoft corporation under windows area but it is idea is to avoidance the mine and one person can play it in the same computer but my game idea is opposite the game of Microsoft, The main idea (in my case) revolves around finding the mine in the area of the game. Two persons can play the game on two computers through the network by the mobile agent, as illustrated in the following algorithm.

Procedure Mine_Sweeper_Manager (IP, PortNumber) \\ used to Control the Mine Sweeper on the remote computer

```
step1, step2, step3 as Boolean
ClickID as integer              \\ number of click in any position on mine sweeper plate.
```

```
Portnum as integer                    \\ port to scan through
Password as string                    \\ password of the agent

Begin
step1= Open (Portnumber)                    \\ To Check the port is Open or not
step2= Check (IP, portnm)                   \\ To Check the Computer Online or offline
step3= Check_Pass (Password)                \\ to check the password of the agent
If (step1 and step2 and step 3) = true then
Start_ Mine_Sweeper (IP, PortNumber, clickID)     \\ start Mine Sweeper window on
                                                   remote computer.
End__white_board (IP, PortNumber, clickID)        \\ end Mine Sweeper window on
                                            remote computer.
Count_grade (IP, PortNumber, clickID)       \\ count grade for each player
If (Count_grade) =23 then             \\ compare the two grads with 23 to one person to win and others
                                      lost to end the game.
Else    \\ for compare grade
Continue playing game until one person wins the game.
Else
Goto finish
End if
: finish

End          \\ End of (Mine_Sweeper _Manager) procedure
```

The small (MAP) system that we have built is meant to help the user by showing him all the connected computers around, and to help him close and check the open port, check the online and offline computers, send messages to computers, manage and control the one or more selected computers, and communicate with the computer(s) selected -more than any other application or tool in the network area would do.

Chapter 5

Conclusions and Future Works

5.1 Conclusions:

Mobile Agent systems are complex software entities whose behavior, performance and effectiveness cannot always be anticipated by the designer. Their evaluation often presents various aspects that require a careful, methodological approach as well as the adoption of suitable tools, needed to identify critical overheads that may impact the overall system performance, stability, validity and scalability. In this thesis, we have used the mobile agent technology to establish communications between local networks, and we have developed a small (MAP) system tools that uses the mobile agent, which navigates through the current network and searches in the routing table about the computer that registered in it, and returns the IP and the name of computers that registered in it to check the status of these computers, whether it is online or offline, and to check the open ports for these computers.

Mobile agent professional systems are used mainly to control two or more small LAN and manage the computers that registered in it, that having mobile agent resides in it, and we can control all component of these computers and the user can't feel with any changing on the computer by mobile agent, which helping us to control the computer by easy and efficiency way and we protect the mobile agent from removing from remote computer by changing the name of that mobile agent, and changing the port number and using password with mobile agent to protect it from any attack that can change the objective of mobile agent and destroy the remote computer.

We use (MAP) for distributed systems by split the computers in the LAN, to two or more groups that each group can controlled by central server and each servers can communicate with others servers by sending a messages between them to reach to the more scalable architecture using the mobile agent that can travel from group to others.

The efficiency of (MAP) Tools depends on the local network architecture and infrastructure. We experiment (MAP) Tools on the local area network that contains 40 computers in it , and the speed of the connection is 100 Mps, with different operating systems installed on these computers, and we experiment these tools on network of class **C** and on network of class **B** and this experiment was done successfully, and we make another two experiments; the first to experiment these tools on wireless network and it is done successfully, and the second experiment is to experiment the operation of sending the mobile agent through a Bluetooth technology to more than one computer that have Bluetooth device installed on it, to receive this mobile agent .This experiment was done successfully.

From the IP which we received from all computers around to the current computer we have built the set of the following important services which is listed in table 6.1 and all of them are implemented and experimented successfully on (LAN) of class **C** and class **B**:

Table 6.1: The Main and Important Tools

Services Name	Function
Mobile IP Checker (to get the IP)	Return all IP in selected class networks and the name Of the Computer That the IP belong to it
Mobile Agent Scanner (to find the open ports)	Return all the open port in your device directly or To other Device or computer Remotely Using the mobile agent
Mobile Agent Checker for Online/Offline Computers	Return All IP That Online Or Offline in selected saved List
Mobile agent services Messages	send the message to the selected computer by The IP of the Computer Or By the computer Name

Table 6.1: The Main and Important Tools (Cont.)

Class Name Recovery	Used To Return The Class Name And other Component of selected IP
Process Controller	Control The Process Of Remote Computer By New Process Or Kill It
File Browser controller	Control The File Of Remote Computer by upload or download file and control property of these files.
Information Controller	Facilities you can get all the Information about the selected computer Through it.
IP/Computer Name Recovery	Retrieve The IP for The Selected Computer Name Or Computer Name for the Selected IP
Mobile agent special Messages	Send Different messages Such As critical message or question message or Information message and other type

5.1.1 Advantage of (MAP):

1. (MAP) takes an advantage over (RPC) because it doesn't require any privilege on remote device because it uses the IP to control the remote device.

2. Unlike remote procedure calls, where a process invokes procedures of a remote host, process migration of (MAP), allows executable code to travel and interact with databases, file systems, information services of another computer and other mobile agents.

3. We used the password to protect the mobile agent from any attack from any other person to change the behavior of mobile agent and objective of it.

4. We used different ports number to control many computers at the same time through the mobile agents to increase the efficiency of these tools.

5. We can know by using these tools which mobile agent with which port number that connection of it is disconnected.

5.1.2 Limitation of (MAP):

1. The efficiency of (MAP) tools depend on local area network architecture and the speed of the connection of that network that using mobile agent over it.

2. Some of tools in (MAP) need some infrastructure in the computer that running on it, for the server device that controls other mobile agent.

3. Some of tools in (MAP) need some infrastructure in the remote computer that having the mobile agent resides in it.

5.2 Future works:

Using the mobile agent in communicating between local networks is still due to an open research topic. Our thesis –just like other many other researches- has many points open for study and improvement. As a future work, we recommend the following points:

- To apply the communication through (Wide Area Networks) WAN.
- To use the mobile agent in the diagnosis of processes of remote computers.
- To develop a complete control system to detect and analyze the entire agent on the remote computer at the same time.
- To apply the proposed algorithms for the detection and analysis of other parts in the computer networks filed, using the mobile agent technology.
- To increase the efficiency and speed of IP algorithms through using the multi agent system.

References:

Awerbuch, B. and Peleg, D. (1995)." **Online tracking of mobile users"**. Journal of the ACM, 42(5):1021–1058, September 1995.

Ballintijn, G., Steen, M. V., and Tanenbaum A. S. (1999)." **Lightweight crash recovery in a wide-area location service"**. In Proc 12th Conference on Parallel and Distributed Computing Systems, August 1999.

Birman, K. P., and Joseph, T. A., (1987). **"Exploiting virtual synchrony in distributed systems"**. In Proc 11th ACM Symposium on OS Principles. ACM, November 1987.

Bohoris,C., Liotta, A., Pavlou,G.(2000)." **Software Agent Constrained Mobility for Network Performance Monitoring"**, Proc. of the 6th IFIP Conference on Intelligence in Networks (SmartNet 2000), Vienna, Austria, ed. H.R van As, pp. 367-387, Kluwer, (September 2000).

Cheng, D. T. and Covaci, S., (1997). **"The OMG Mobile Agent Facility"**: A Submission, in Rothermel, K. and Popescu-Zeletin, R., Eds., Mobile Agents, Springer-Verlag, 1997.

Chess, D., Harrison, C., and Kershenbaum, A. (1997). **"Mobile agents: Are they a good idea?"** In J. Vitek and C. Tschudin, editors,*Mobile Object Systems Towards the Programmable Internet*,Volume LNCS 1222, 1997.

Demmer ,M. J. , and Herlihy, M. P.(1998)." **The arrow distributed directory protocol"**. In Proc 12th Symposium on Distributed Computing, volume LNCS 1499, 1998.

FIPA. FIPA ’98, (1998).**"Draft Specification"**: Part 1 Agent Management. (1998).

Glass, G., (1999). **"ObjectSpace"**, Overview of Voyager: ObjectSpace’s Product Family for State of-the-Art Distributed Computing. White paper, ObjectSpace, (1999). Available on web: http://www.objectspace.com/products /documentation/ VoyagerOverview.pdf.

Goldszmidt ,G., Yemini, Y. (1998)." **Delegated Agents for Network Management"**. IEEE Communications Magazine, Vol.36 No.3, (March 1998).

Goldszmidt, G. and Yemini, Y. (1995). **"Distributed Management by Delegation"**. In Proc. of the 15th International Conference on Distributed Computing Systems, pages 333–340, Los Alamitos, CA. IEEE Computer Society, IEEE Computer Society Press.

Goldszmidt ,G.(1993)." **Distributed System Management via Elastic Servers"**. In Proceedings of the IEEE First International Workshop on System Management, Los Angeles, California, (April 1993).

Goldszmidt ,G., and Yemini, Y. (1996).**"Delegated Agents for Distributed System"**. Management". IFIP/IEEE DSOM 1996 Workshop. L'Aquila, Italy، (October 28-30th 1996)

Goldszmidt ,G., (1996).**"Distributed Management by Delegation"**. PhD Thesis،Columbia University, New York, (1996).

Green, S., Hurst, L., Nangle, B., Cunningham, P., Sommers, F. and Evans, R., **"Software Agents"**: A review, Technical Report, Department of Computer Science, Trinity College, Deblin, Ireland.

Holliday, M.A.(2003). **"Animation of computer networking concepts"**. ACM Journal of Educational Resources in Computing 3 (2003) 1–26.

Jul, E., Levy, H., Hutchinson, N. and Black, A. (1988).**"Fine-grained mobility in the Emerald system"**. ACM Transactions on Computer Systems, 6(1):109–133, February 1988.

Keidar,I. , Sussman,J., Marzullo,K. and Dolev,D.(1999).**" A client-server oriented algorithm for virtually synchronous group membership in WANs"**. Technical Report CS1999-0623, University of California, San Diego, 1999.

Lawrence, A. (1995). **"Agents of the Net"**. New Scientist, 147(1986):34–37.

Liotta ,A., Ragusa,C., and Pavlou ,G.,(2002).**"Running Mobile Agent Code over Simulated Inter-networks:an Extra Gear towards Distributed System Evaluation"**,2002

Liotta ,A., Ragusa,C., and Pavlou ,G., (2002).**"Exploiting Agent Mobility for Large-Scale Network Monitoring"**, 2002.

Liotta ,A., (2001).**"Towards Flexible and Scalable Distributed Monitoring with Mobile Agents"**, Ph.D. thesis, Dept. Comp. Sci., Univ. College London, U.K., July 2001.

Loke, S.W., Rakotonirainy, A., and Zaslavsky, A. (2000). **"Enabling Awareness in Dynamic Mobile Agent Environments"**. (Short paper). Proceedings of the 15th Symposium on Applied Computing (SAC 2000), Como, Italy, March 2000,ACM Press.

Meyer ,K., Erlinger, M., Betser, J., Sunshine, C., Goldszmidt, G. and Yemini Y.(1995), **"Decentralising Control and Intelligence in Network Management"**. Proceedings of the 4th International Symposium on Integrated Network Management, Santa Barbara, CA, (May 1995).

Moreau, L. (1999).**"Distributed directory service and message router for mobile agents"**. Technical Report ECSTR M99/3, University of Southampton, 1999.

Mountzia ,M.A., and Dreo-Rodosek, G. (1996).**"Delegation of Functionality: Aspects and Requirements on Management Architectures"**. Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM'06), (October 1996).

(online) Ad Astra Engineering Inc. Jumping Beans - The Mobility Framework. web:http://www.JumpingBeans.com, (1999).

Padovitz ,A., Zaslavsky, A. and Loke, S. W. (2002).**"Awareness and Agility for Autonomic Distributed Systems: Platform-Independent Publish-Subscribe Event-Based Communication with Mobile Agents"**,(2002).

Pawel, T., and Wojciechowski. (2000).**"The Nomadic Pict System"**, 2000.Available electronically as part of the Nomadic Pict distribution. http://lsewww.epfl.ch/pawel/nomadicpict.html.

Pawel, T., and Wojciechowski. (2000).**"Algorithms for location-independent communication between mobile agents"**,Swiss Federal Institute of technology (EPFL), operating systems laboratory ,CH-1015 Lausanne,2000.

Pawel, T., Wojciechowski, and Peter Sewell (2000). **"Nomadic Pict: Language and infrastructure design for mobile agents"**. *IEEE Concurrency*, 8(2):42–52, April-June 2000.

Pawel, T.,and Wojciechowski. (2000) *"***Nomadic Pict: Language and Infrastructure Design for Mobile Computation"**. PhD thesis, University of Cambridge, 2000. Also appeared as Technical Report 492, Computer Laboratory, University of Cambridge, June 2000.

Sewell, P., Pawel, T., Wojciechowski and Benjamin, C. (1999). **"Location-independent communication for mobile agents: A two-level architecture"**. In Henri E. Bal, Boumediene Belkhouche, and Luca Cardelli, editors, *Internet Programming Languages*, LNCS 1686, 1999. Also appeared as Technical Report 462, Computer Laboratory, University of Cambridge, April 1999.

Steen, M. V., Hauck, F. J., Ballintijn, G., and Tanenbaum A. S. (1998). **"Algorithmic design of the Globe wide-area location service"**. The Computer Journal, 41(5):297–310, 1998.

Straber ,M., and Schwem, M. (1997). **"A Performance Model for Mobile Agent Systems"** ‹H. Arabnia (Ed.), Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97), Vol. II‹CSREA, pp. 1132-1140, 1997.

Vinoski, S., (1997). **"CORBA overview: CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments"**, IEEE Communications Magazine, Vol. 14, No. 2, Feburary, 1997.

White, T., and Pagurek, B. (1998), **"Towards Multi-Swarm Problem Solving in Networks"**. In Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS ’98), July, 1998.

Yemini, Y., Goldszmidt, G. G.,and Yemini S. (1991).**"Network Management by Delegation"**. Integrated Network Management II, Amsterdam (1991).

تَحسين دور الوكيل النقَّال في تَأسيس الاتصال بين الشبكات المحليّة

إعداد

نبيل مصطفى العساف

المشرف

الدكتور عماد خالد صلاح

ملخص

لعِدّة سَنَوات إلى الآن،إن تطبيق وفائدة تقنيات الوكيل النقّال للنظام المُوَزَّع وإدارة الشبكة(إس إن إم) قد عُرِفـا. إن إحدى النقاط الرئيسية هي أنْ تُسنَدْ إلى الوكلاء المستقلين ذاتياً والنقَّالين احتمالية أن يتم إسناد دور لهم في الإدارة. ومع ذلك وبحد ذاته، فإن الشبكة وحساب الحِمّل عليها أصبح مُوَزَّعا بدلاً مِنْ أن يكون مُركَز نحو وعلى مدير المضيّف وحده.

أن ترتيبات الشبكة من المهام الأخرى التي يُمْكِنْ أنْ يكون لها بَعْض التأثير أيضاً على الهندسة الشكلية الفعّالة للشبكة الكاملة المُدَارة. إن استعمال أيّ وكيل نقّال أساسا لإدارة الشبكة والنظام يُشيران لأول خطوة في نَشْر البناء التحتي (حيث يجب أن يكون هناك عملاء لكي يستضيفوا الوكلاء النقّالين). ثانياً، أنهُ قادر على بَرْمَجَة المخططات المتكوّنة من النظام أو قادر على شبك العناصر التي يجب أنْ يَزُوروها الوكلاء النقّالين لكي تُنفذ مهام إدارتِهم.

في هذه الأطروحة، أقدّم خوارزميات أكثر فعّالية التي تساعدني لاستعمال الوكيل النقال للاتصال مع الشبكات المحليّة بالعديد من الأدوات التي تجد كل آي بي لكلّ الحواسيب في نفس النطاق أو خارجه واسم ذلك الحاسوب الذي سجّل بواسطته على الشبكة وأيضا أقدم الخوارزميات التي تستعمل الوكيل النقال لإيجاد كل الأبواب

المفتوحة للحاسوب البعيد وتحصل على حالة الحاسوب في الشبكة المختارة لرؤية كل الحاسبات المتصلة أو غير المتّصلة لكي اتّصل معها وأنا أقدّم خوارزميات تستعمل الوكيل النقال كعقدة ومركز للسيطرة على حاسوب بعيــد وإدارته وأحصل على كلّ الملكيات والأجهزة الخاصة به وهناك خوارزميات أخرى مشروحة وموضحة بالتفصيل في هذه الأطروحة.