Mobile Agent- A Novel Paradigm for Data Mining

A thesis
Submitted in the partial fulfillment of the
requirement for the award of degree
of
Master of Engineering
in
Software Engineering

Submitted By:
HARVENDRA KUMAR
(80731029)

Under the supervision of:
Dr. Anil Kumar Verma
Mr. Ajay Kumar

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

MAY 2009

CERTIFICATE

I hereby certify that the work which is being presented in the Thesis entitled, **"Mobile Agent- A novel paradigm for Data Mining",** in partial fulfilment of the requirements for the award of degree of Master of Engineering in Software Engineering (SE) submitted in Computer Science and Engineering Department (CSED) of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of ***Dr. Anil Kumar Verma and Mr. Ajay Kumar.***

The matter presented in this Thesis has not been submitted for the award of any other degree of this or any other university.

Harvendra Kumar

(80731029)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Anil Kumar Verma)

Computer Science and Engineering Department

Thapar University

Patiala

(Mr. Ajay Kumar)

Computer Science and Engineering Department

Thapar University

Patiala

Countersigned by

(Dr. SEEMA BAWA)

Professor & Head

Computer Science & Engineering.

Department

Thapar University

Patiala.

(Dr.R.K.SHARMA)

Dean (Academic Affairs)

Thapar University,

Patiala.

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my advisors **Dr. Anil Kumar Verma and Mr. Ajay Kumar,** faculty, Computer Science & Engineering Department for their immense help, guidance, stimulating suggestions and encouragement all the time. They always provide a motivating and enthusiastic atmosphere to work with, it was a great pleasure to do this thesis under their supervision.

I would like to thank **Dr. Seema Bawa**, Head, Computer Science and Engineering Department, Thapar University, Patiala for providing necessary facilities in the Department, to work. The most valuable thing, I acquired from my alma mater Thapar University is to protect, survive and endure myself in this world with increased confidence and additional faith in my abilities to achieve. I am also thankful to the authors whose works I have consulted and quoted in this work.

I am very thankful to my respectable family for their fortitude. Nothing would have ever been possible without my parent's unconditional love and encouragement. Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Harvendra Kumar

(80731029)

ABSTRACT

The mobile agent paradigm has revolutionised the distributed computing environment. There are different approaches used in distributed computing. These are - client-server architecture, remote procedure architecture, code on demand architecture and mobile agent architecture. The client-server is based upon the concept of a server, that serves the various request of the clients and in remote procedure call approach, the machine can connect to another machine and retrieve the information remotely. The mobile agent technology is built upon the advancement in computing and communication technology. Further, it suits homogeneous as well as heterogeneous networks either using a wired or wireless media. To define it in simple words, we can say that - the mobile agent is an autonomous program that can migrates under its own or host control in the networks. Thus, it is simple, light in weight and performs the desired task efficiently, moreover this paradigm does not load the network and also the memory requirement is low. There are different programming languages used for writing programs that incorporate the mobile agent paradigm. Each of these languages has certain characteristics, which make them more suitable for a certain type applications that make use of mobile agent paradigm. Keeping in view the advantages and limitations of mobile agents it is beneficial for us to use the mobile agent paradigm.

Data Mining is a process in which data can be extracted from different data warehouse based upon some pattern. Data Mining allows users to analyze data from many different perspective or angles, categorize it, and summarize the relationships.

PMADE (Platform for Mobile Agent Distribution and Execution) is framework for large-scale multi-agent systems. The work presented makes use of mobile agent paradigm on PMADE platform for the process of data mining. The mobile agent paradigm is implemented on this platform and the process of data mining is performed and the desired output is presented.

Keywords: – Distributed Computing, Mobile agent, Data Mining, PMADE platform.

Abbreviation: -

MA	MA Mobile Agent
CS	Client-Server
REV	REV Remote evaluation
DG	Data Gathering
DM	Data Mining
DKD	Distributed Knowledge Discovery
DDM	Distributed Data Mining
IRS	Information Retrieval Systems
MAF	Mobile Agent Facility
OMG	Object Management Group
P2P	Peer to Peer
AH	Agent Host
AS	Agent Submitter
PDGA	Pattern Database Generator Agent
PDPA	Pattern Database Processor Agent
PBSA	Pattern Based Search Agent
PBPA	Pattern Based Processor Agent
PMADE	Platform for Mobile Agent Distributed and Execution

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

CHAPTER 1
INTRODUCTION

A computer network is a set of computers or devices connected to each other with the ability to exchange data. Network users are able to share files, printers, and other resources; send electronic messages; and run programs on other computers. Computer network play an important role in today's life. It would be appropriate to learn the world as "global village", because of computer network. Without networks almost all communication in the world would cease. Modern amenities like- televisions, telephone, the Internet, etc are available today due to computer network.

Types of connection of network:-

- **Physical connection:** Computers directly transmit and receive signals. Physical connections are defined by medium used to carry the signal, the geometric arrangement of the computers and the methods used to share information.
- **Logical, or virtual:** connections that allow computer applications, such as e-mail programs and the browsers used to explore the World Wide Web (WWW), to exchange information. Logical connections are created by network protocols and allow data sharing between applications on different types of computers. Some logical connections use Client-Server [24] application software and some are primarily for file and printer sharing.

1.1 Components of Network layer: -

- Application software
- Network software
- Network hardware

Application software: - consists of computer programs that interface with network users and permit the sharing of information, such as files, graphics, and video, and resources, such as printers and disks. One type of application software is called Client-Server. The Client computers send requests for information or requests to use resources to other computers, called Servers that control data and applications. Another type of application software is

called peer-to-peer, in which the computers send messages and requests directly to one another without a server intermediary.

Network software: - consists of computer programs that establish protocols, or rules, for computers to talk to one another. These protocols are carried out by sending and receiving formatted instructions of data called packets. Protocols make logical connections between network applications, direct the movement of packets through the physical network, and minimize the possibility of collisions between packets sent at the same time.

Network hardware: - Network hardware is made up of the physical components that connect computers. Two important components are the transmission media that carry the computer's signals, typically on wires or fibber-optic cables, and the network adapter, which accesses the physical media that link computers, receives packets from network software, and transmits instructions and requests to other computers. Transmitted information is in the form of binary digits, or bits (1s and 0s), which the computer's electronic circuitry can process.

1.2 Data Gathering or Data mining: -

Data gathering (DG) or data mining (DM) is the process of analyzing data from different perspectives and summarizing it into useful information. DM information can be used to increase the revenue, cuts costs, or both. DM allows users to analyze data from many different perspective or angles, categorize it, and summarize the relationships. DM is not a new term. Technically, it is the process of finding the correlations or patterns among dozens of fields in large relational databases. Companies have used powerful computers to filter through volumes of supermarket scanner data and analyze market research reports for years. However, continuous innovations in computer processing power, disk storage, and statistical software are dramatically increasing the accuracy of analysis while driving down the cost.

1.3 Computational Paradigms:-

There are different computational paradigms proposed by the researcher community, the four main paradigms are :-

- Client-Server paradigm
- Remote Evaluation paradigm
- Code on demand paradigm and
- Mobile agent paradigm

The Client-Server is based upon the concept of a server, which serves the various request of the clients and in remote evaluation approach, a machine can connect to another machine and retrieves the information remotely. The mobile agent technology is built upon the advancement in computing and communication technology over the wired and wireless networks. Further, it suits homogeneous as well as heterogeneous networks. To define it in simple words, we can say that - the mobile agent is an autonomous program that can migrates under its own or host control in the networks. Thus, it is simple, light in weight and performs the desired task efficiently, moreover this paradigm does not load the network and also the memory requirement is low.

The appearance of fast wide-area networks has moved distributed computing to a new area in which distributed resources in a wide-area network are more closely coupled to provide a single integrated platform for computing and data storage, viz., peer-to-peer (P2P) networks, which tie together the computing power of hosts in a network and make their under-utilized resources available to users. P2P systems are formed from dynamic connections among autonomous computer nodes, producing large and complex application-layer overlay networks. Informally, P2P systems can be described to be distributed systems in which all nodes are peers in the sense that they have equal roles and responsibilities. The nodes in the distributed system have identical capabilities and responsibilities, and all communication is symmetric. P2P systems are characterized by decentralized control, large scale, and extreme dynamics of their operating environment. P2P systems are used for file sharing (i.e., exchanging files between peers) and file storage (i.e., the P2P network is used as a distributed file system).

The appeal of mobile agents is quite appealing - mobile agents roaming on the Internet could search for information, find us great deals on goods and services, and interact with other agents that also roam networks (and meet in a gathering place) or remain bound to a particular machine. Significant research and development into mobile agent has been conducted in recent years, and there are many mobile agent architectures available today such as PMADE, Ajanta, Odyssey, Aglets and etc.

Platform for Mobile Agent Distribution and Execution (PMADE) is a framework designed to support large-scale multi-agent systems. The main reason behind the use of PMADE is that it supports large scale multi agent systems & simplifies the application development and deployment by freeing the application developer of all low-level details

including communication, security and scheduling. In this proposed work, gathering data can be form of such files- doc files, pdf files, text files, and etc.

In the proposed work, we have used mobile agent paradigm to gather the data from different hosts presents on the network.

CHAPTER 2
COMPUTING PARADIGMS

From the network perspective [9], there are additional challenges and problems in meeting bandwidth requirement. To overcome this problem; new techniques, languages and paradigms have evolved, which facilitate the creation of such applications. The various computing paradigms [1] can be listed as below:-

- The Client - Server (CS) paradigm
- The Remote Evaluation (REV) paradigm
- Code on demand paradigm
- The Mobile Agent (MA) paradigm

2.1 Client - Server (CS) paradigm: - In the Client-Server (CS) paradigm [24], data processing mainly takes place in the host client. In fact, the work of the server is limited to execution of some basic procedures for the data retrieval and storage. Before being sent to the Client, data only undergo a soft initial filtering. The host server behaves as a simple remote storage system. Together with all of the other servers and the interconnection network, makes the whole system to form a “big repository" of information available to the different clients. The server usually makes available some procedures for handling the stored data which are designed for responding to criteria of general effectiveness. The actual data processing is therefore left to the host client, where the user can execute procedures for the kind of processing desired. This type of CS scheme is used when we want to create a very simple system from the management point of view, or structures with a high level of security.

In fact, the procedures for accepting the requests and selecting the data embedded into the servers. An advantage of this architecture is the possibility of controlling the type and the ways of access to the data stored in the server. Consequently, the level of security is very high. In fact, if the user has specific requests concerning the modes of data processing, and if the server (or better its manager) does not provide for that specific type of operations, the only possibility commits in retrieving much more data than needed, and then to perform the operations of processing and selection in the client. In these cases, the server provides a huge amount of documents, in order to assure a wide basis of selection. Of course, all that causes an overload of both the server and the communication system. In fact, the amount of data

exchanged may be considerable. Consequently, the host client must have its own processing capability. Of course, this cannot be low.

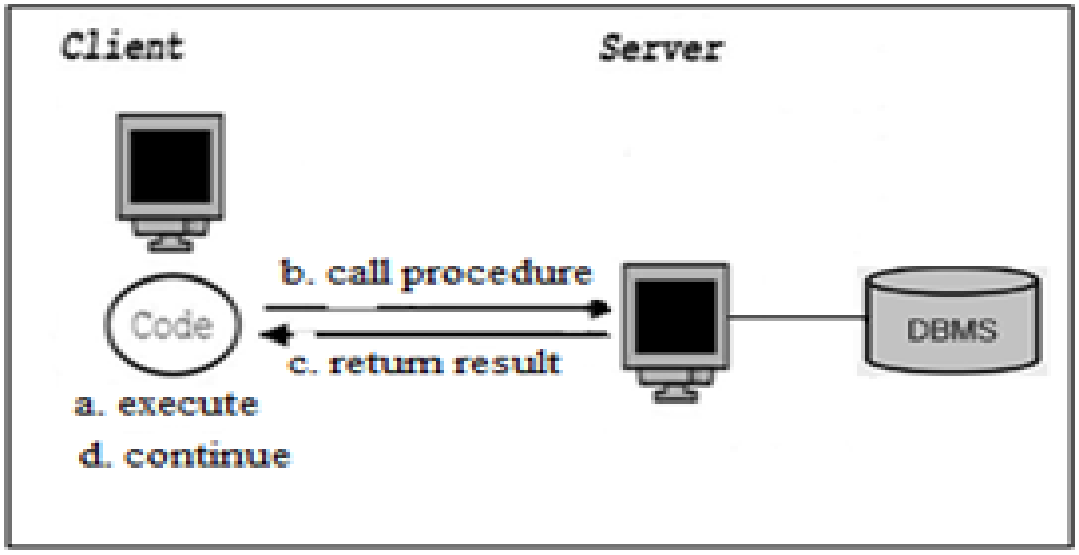


Figure 2.1: Client-Server Architecture [24]

In the Client-Server paradigm, as shown in Figure 2.2, a server advertises a set of services that provide access to some resources (such as database). The server hosts the code that implements these services locally i.e. server holds the know-how. Finally, it is the server itself that executes the service and thus has the processor capability. If the client is interested in accessing a resource hosted by the server, the client will simply use one or more of the services provided by the server. Client needs some "intelligence" to decide which of the services it should use. The server has the know-how, resources, and processor to serve the various needs of the Client.

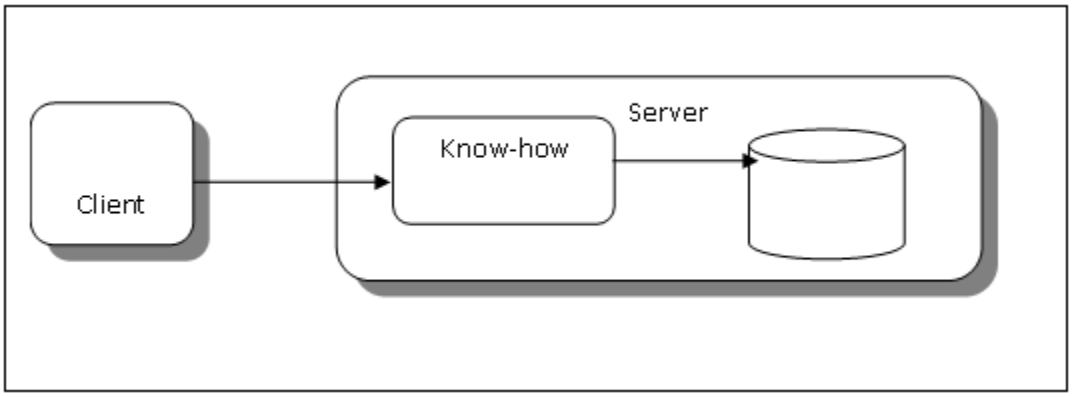


Figure 2.2: Client-Server paradigm [38]

Most distributed systems are based on this Client-Server paradigm. A wide range of technologies such as remote procedure calling, common object request broker agent (CORBA), and Java remote method invocation (RMI) [30] support this technology.

2.2 Remote Evaluation paradigm: - Unlike the typical Client - Server, the paradigm of Remote Evaluation (REV) paradigm [24] implies that server receives not only the processing requests from the client, but also the whole code needed for performing operations of selection on the data stored. The response of the server is therefore limited to sending the information that can be actually used and required by the client, with no additional overhead.

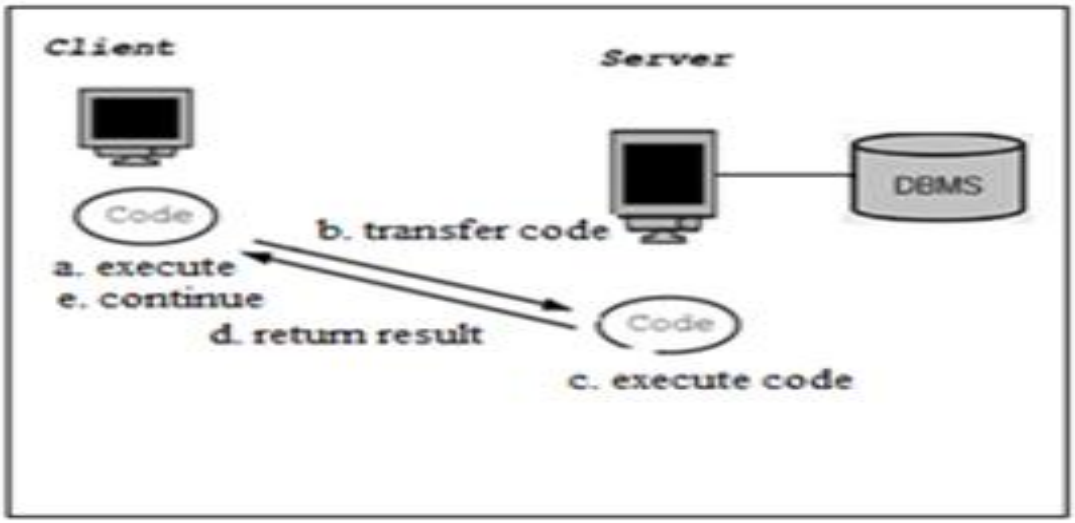


Figure 2.3: Remote Evaluation Architecture [24]

Besides, since the user can use a customized code in the server, the data sent in output are ready for the use, and they only need negligible additional processing. From this point of view we can also think of an environment with host clients equipped with minimum processing potentialities. The initial cost is therefore higher in comparison with the CS paradigm, and is localized in the opening stage of sessions. In fact, the code for the data processing can be of considerable size and we can easily assume that its size is higher than a simple retrieval request. Of course, this cost is counterbalanced by the reply stage (transmission of search results from the server to the client), because the amount of data passing through the network is more limited. During the stage of design, a system with Remote Evaluation [2] must be created by considering more detailed aspects in comparison with the CS. In fact, the processing architecture of the different servers must be similar (or very well known), so that the code sent by the client can be easily executed on all of the hosts. From this point of view, we can think of a common platform for code execution. This also implies the need for creating protection elements that could assure a high level of security.

2.3 Code on Demand Paradigm: - In the code-on-demand paradigm Figure 2.4, know-how is achieved when needed. Suppose that a client is initially unable to execute its task because of a lack of code (know-how), then, a host in the network provides the needed code. Once the client receives the code, the computation is carried out in the client. The client holds the processor capability as well as the local resources. In contrast to the classical client-server paradigm, the client does not need preinstalled code because all the necessary code will be downloaded.

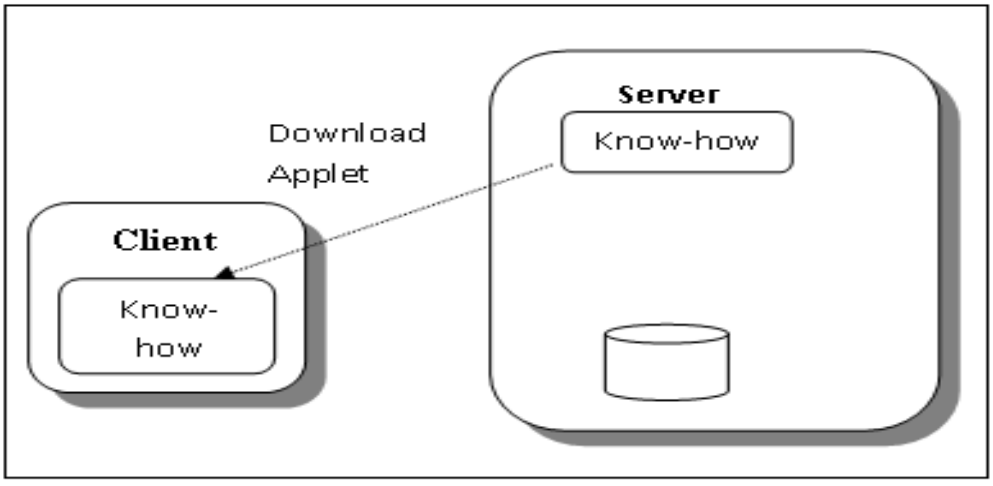


Figure 2.4: Code on demand paradigm [38]

We say that the client has resources and processor, and the host has the know-how.Java applets and servelets are excellent practical examples of this paradigm. An applet gets downloaded in Web browsers and executes locally, whereas servlets get uploaded to remote Web servers and execute there.

2.4 Mobile Agent (MA) paradigm: - A Mobile Agent (MA) [24] is an executable code that can move from a host to another, either according to decisions taken independently or according to outside parameters or after some interactions with other agents. The context of the program is transferred, as well as the code, during the migration. This way, there is a kind of suspension of the execution of the program, waiting for the subsequent resume state on a remote machine. The system of Remote Evaluation is a more limited approach than the MA. In fact, a code migration is present in the REV, but there is always a direct interaction between the client and the server. This means that the code sent by the client returns the data directly to the source. Besides (when this operation is done), the process is completed, so the context of execution of the program is limited to the single host. Conversely, the mobile agent system can be used for performing the research operations more effectively. In fact, the agent has the procedures for operating on the database according to the ways desired by the user, and can also make independent decisions, such as migration to other sites or returning the results obtained to the user, if they are considered sufficient. In this sense, the interaction between the user and the agent is limited to the stages of transmission and return of data. What takes place within this time limit depends only on the way the agent was designed.

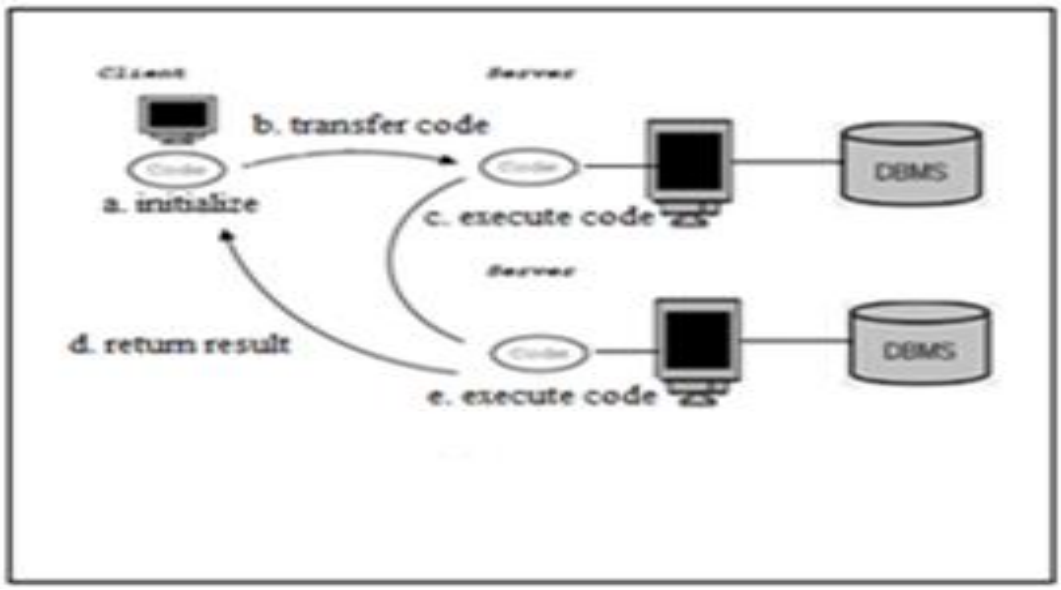


Figure 2.5: Mobile Agent Architecture [24]

So, we can say that the user loses the control of the agent, once it has been sent. This feature implies two direct consequences: -

- The size of the code of the agent is higher than the one needed for a simple REV (the structure of an MA is more complex!)
- When the agent migrates, it has to carry the partial results of the processing performed in the hosts visited before. We can therefore expect that the amount of data transferred in each migration tends to increase. The agent can decide to limit the data considered interesting for the user dynamically, even by discarding the data selected in the previous hosts. Agents with a maximum quota of user data, which can be moved in each migration, can therefore be designed.

A Mobile Agent (MA) as shown in Figure 2.5 is an autonomous transportable program (or object) that can migrate under its own or host control from one node to another in a heterogeneous network. In other words, the program running at a host can suspend its execution at an arbitrary point, transfer itself to another host (or request the host to transfer it to its next destination) and resume execution from the point of suspension. When the agent reaches a server, it is delivered to an agent execution environment. Then, if the agent possesses necessary authentication credentials, its executable parts are started. To accomplish its task, the mobile agent can transport itself to another server in search of the needed resource/service, spawn new agents, or interact with other stationary agents. Upon completion, the mobile agent delivers the results to the sending client or to another server.

Once it is launched by a user, can travel from node to node autonomously, and can continue to function even if the user is disconnected from the network. They implement a computational metaphor that is analogous to how most people conduct business in their daily lives: visit a place, use a service, and then move on. It behaves like a human agent, working for clients in pursuit of its own agenda [16, 17, 18]. Various attributes of a MA are identification to identify a MA and its dispatching station, itinerary consisting of the number and order of the hosts to be visited by MA, data unit containing the required data, code unit containing the transportable code, execution state, i.e., output of the serialization process [19], and Mobile Agent paradigm of the distributed computing is different for other paradigms. In other paradigms, independent processes collaborate by exchanging data over

their network links. The MA, a process is transported, carrying with it the shared data as they visits individual processes on thier itinerary. The following Figure 2.6, highlights the different approaches made are in a MA environment and conventional distributed paradigms processing environment.

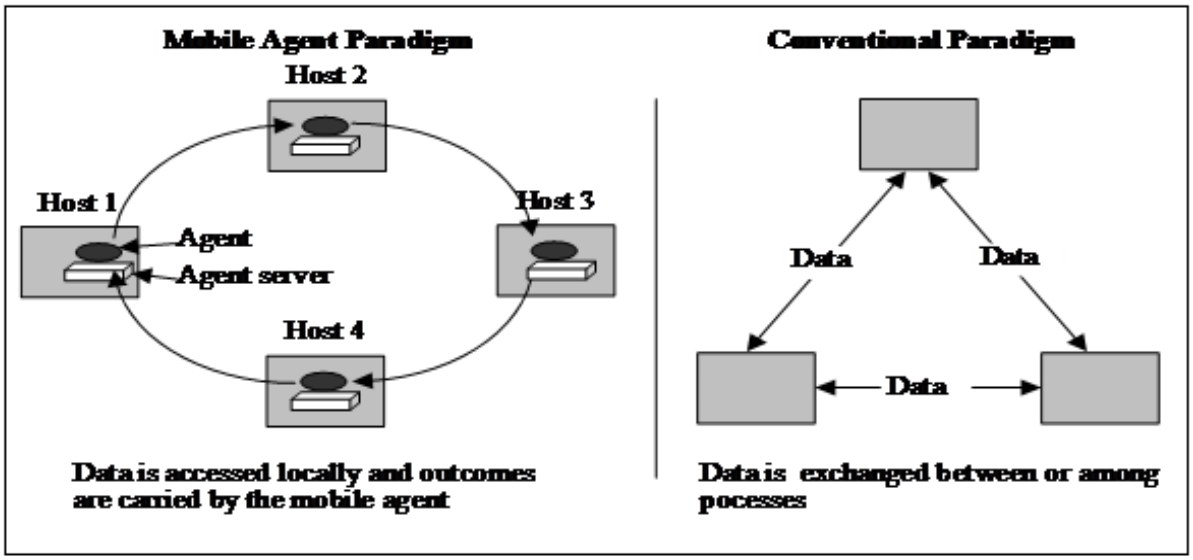


Figure 2.6: Mobile agent vs. conventional distributed paradigms [22]

In the above Figure 2.6, very clear the difference between Mobile agent and conventional distributed paradigms. In conventional paradigm only the data transfers during the information retrieval that's why very much load on the network will be when transferring data, and in Mobile agent paradigm, only result transferred during the information retrievals, load on the network will be less.

It is clearly evident from the discussion above that MA is a novel paradigm and offers melodious advantages over conventional distributed paradigm. So, it is beneficial to exploit this paradigm.

CHAPTER 3
MOBILE AGENT PARADIGM

The basic concept of an agent is that an agent functions by allowing people to delegate work for them. Software agents [4] are programs that assist people and act on their behalf. A software agent is a computer program whose execution depend upon events and data conditions in its environment and is not under continuous, direct control of a human user. Agents can be divided into two categories:-

- **Stationary Agent: -** A stationary agent executes only on the system where it begins execution. If it needs information that is not on that system or needs to interact with an agent on different system, it typically uses a communication mechanism such as Remote Procedure Call (RPC).
- **Mobile Agent:** A mobile agent is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another. The ability to travel allows a mobile agent to move to a system that contains an object with which the agent wants to interact and then to take advantage of being in the same host or network as the object.

3.1 Mobile Agent Paradigm: -

A key characteristic of the mobile agent paradigm, as shown in Figure 3.1 is that any host in the network is allowed a high degree of flexibility to possess any mixture of know-how, resources, and processors. Its processing capabilities can also be combined with local resources.

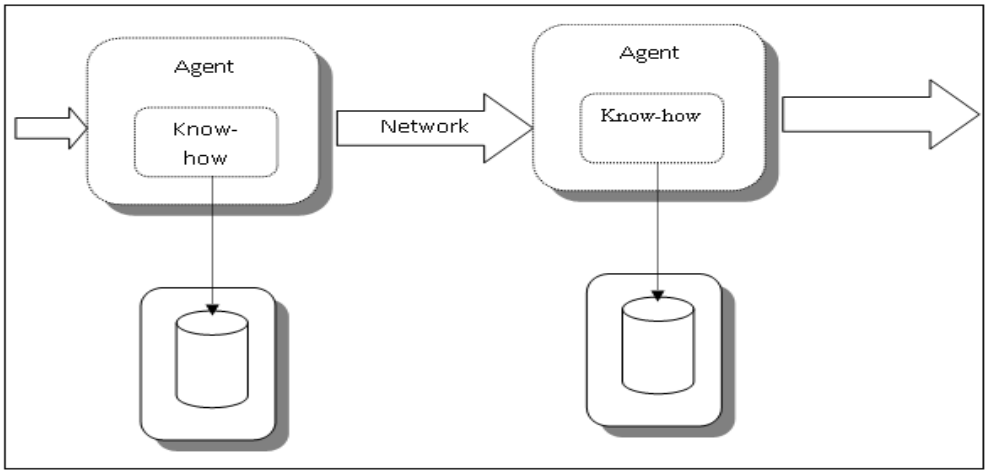


Figure 3 1: Mobile agent paradigm [38]

Know-how (in the form of mobile agents) is not tied to a single host but rather is available throughout the network, where the client and the server have merged and become a host. The applet and the servelet, while serving as client and server extenders, respectively, have been combined and improved with the emergence of mobile agents.

In a nutshell, a MA is a software agent that can migrate, from machine to machine on a heterogeneous network. On each machine, the agent interacts with stationary service agents and other resources to accomplish its task. The agent performs its job wherever and whenever appropriate and is not restricted to be co-located with its client. Thus, there is an inherent sense of autonomy in the mobility and execution of the agent.

3.2 Mobile Agent: - Ongoing research on mobile agent technology [31] has presented a wide range of distributed applications in which a MA can be used effectively and efficiently. Computation paradigm which was came before mobile agent paradigm has lot of deficiency such as- they need more network bandwidth and more network speed. This new technology promises to solve several problems on different fronts. Mobile agents solve the nagging client-server network bandwidth problem, by reducing the frequency of traffic between client and server exploiting the code to data strategy. Agent architecture also answers the most common and frequent problems of irregular or unreliable network connections, by allowing clients to go offline after submitting a job and then receive the results when they re-establish the connection. Applications can introduce mobile agents into a network, allowing them to roam the network either on a predetermined path, or one that the agents themselves determine based on dynamically gathered information. Having accomplished their goals, the agents may either terminate or return to their home site in order to report their results to the user. The Mobile agent paradigm provides various key advantages; as described by Danny Lange's Seven Good Reasons [10] for Mobile Agents:

3.2.1 They reduce the network load: - MAs reduces the network traffic because only shared resultant data is carried over the network and unnecessary intermediate results transmission is reduced. Distributed systems often rely on communications protocols that involve multiple interactions to accomplish a given task. This is especially true when security measures are enabled. The result is a lot of network traffic. Mobile agents allow you to package a conversation and dispatch it to a destination host, where the interactions can

take place locally. Mobile agents are also useful when it comes to reducing the flow of raw data in the network. When very large volumes of data are stored at remote hosts, these data should be processed in the locality of the data rather than transferred over the network. The motto is simple: move the computations to the data rather than the data to the computations.

3.2.2 They overcome network latency: - MAs allow efficient and economical use of communication channels, which may have low bandwidth and high latency. The other conventional distributed computing paradigms require repeated exchange of data over the network links. If a large amount of data is involved, the consumption of network bandwidth can be considerable, with a resultant delay in response time or latency. With mobile agents, a single serialized object is transmitted over the network, potentially reducing the consumption of network bandwidth. Critical real-time systems, such as robots in manufacturing processes, need to respond in real time to changes in their environments. Controlling such systems through a factory network of a substantial size involves significant latencies. For critical real-time systems, such latencies are not acceptable, so MAs offer a solution, because they can be dispatched from a central controller to act locally and directly execute the controller's directions.

3.2.3 They encapsulate protocols: - MAs enable the use of portable, low-cost, personal communications devices such as PDAs, to perform complex tasks even when the device is disconnected from the network. MAs, for example, can migrate off a laptop and roam the Internet to gather information for its user. It can access the needed resources efficiently since it moves to their network location rather than transferring multiple requests and responses across the low-bandwidth laptop connection. Since it is not in continuous contact with the laptop, the agent is not affected by sudden loss of connection, and can continue its task even if the user powers down or disconnects from the network. When the user reconnects, the agent returns to the laptop with the result of its travels. Conversely, an application that lives in the network can send a mobile agent onto the laptop. The agent acts as the application's surrogate, interacting with the user efficiently and continuing to interact even in the event of long-term disconnection.

When data are exchanged in a distributed system, each host owns the code that implements the protocols needed to properly code outgoing data and interpret incoming data, respectively. However, as protocols evolve to accommodate new requirements for efficiency

or security, it is a cumbersome if not impossible task to upgrade protocol code properly. So MAs are able to overcome this problem.

3.2.4 They execute asynchronously and autonomously: - MAs have capabilities of task selection, prioritization, goal-directed behavior, decision-making without human intervention i.e. capability of modifying the way in which they achieve their objectives. Often, mobile devices must rely on expensive or fragile network connections. Tasks that require a continuously open connection between a mobile device and a fixed network probably will embed into MAs, which can then be dispatched into the network. After being dispatched, the mobile agents become independent of the creating mobile device can, and reconnect at a later time to collect the agent.

3.2.5 They adapt dynamically: - MAs have the ability to sense their execution environment and react autonomously to changes. Multiple mobile agents possess the unique ability to distribute themselves among the hosts in the network so as to maintain the optimal configuration for solving a particular problem.

3.2.6 They are naturally heterogeneous: Network computing is fundamentally heterogeneous, often from both a hardware and software perspective. Because MAs are generally computer and transport-layer independent and are dependent only on their execution environment, they provide optimal conditions for seamless system integration.

3.2.7 They are robust and fault tolerant: - MAs can also be used to create programs which can recover themselves in case of a partial failure in the network. When the agents are running on different machines they are vulnerable. One or more machines running on agents can crash or their connection with the other machines can be severed by a network failure. In this case the program will most likely stop functioning and data can be lost, whereas agent is replicated and will not suffer a failure in such a case. This way there is always at least one of each agent type operational so the program will not suffer a failure. The ability of mobile agents to react dynamically to unfavourable situations and events makes it easier to build robust and fault-tolerant distributed systems. Further, if a host is being shut down, all agents executing on that machine will be warned and given time to dispatch and continue their operation on another host in the network.

3.3 Why Java for Mobile Agents: -

MA written in different language such as java, safe-Tcl, Python, scheme48, Obliq, Logic Ware and other. Java is worldwide acceptable language. Write the program once and run anywhere. Since every machine have JVM interpreter. JDK compiler makes byte code. Java language has changed the Web overnight, offers unique capabilities that are fuelling the development of mobile agent systems. Some very good reasons [32] are given below:

- **Platform independence: -** java solves the problem of platform-independent by using byte code. Java byte code is exactly the same on every platform. Java is designed to operate in heterogeneous environments. To enable to Java application to execute anywhere on the network, the computer generates byte code as opposed to non-portable native code.

- **Multithread Programming: -** Multithreaded program enables two or more programs so that they can run parallel with in same application. Each activity performed by own thread in multithreading programming. A thread is a path of execution through the software that has its own call stack and CPU state. Threads run within the context of a process, which defines an address space within which code and data exist, and threads execute.

  - Improve application responsiveness
  - Use multiprocessors more efficiently
  - Improve your program structure
  - Use fewer system resources
  - Improve performance

- **Object Serialization: -** The real power of serialization is the ability of Java programs to easily read and write entire objects and primitive data types, without converting to/from raw bytes or parsing clumsy text data. A key feature of mobile agents is that they can be serialized and de-serialized. Java conveniently provides a built-in serialization mechanism that can represent the state of an object in a serialized form sufficiently detailed for the object to be reconstructed later.

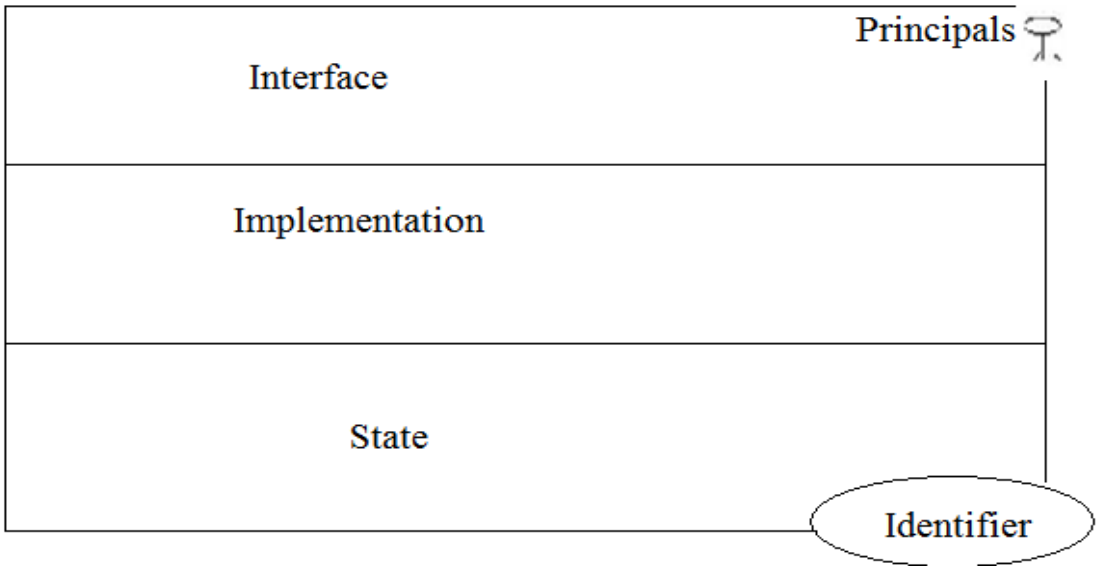
- **Reflection: -** Reflection is commonly used by programs which require the ability to examine or modify the runtime behaviour of applications. This is a relatively advanced

feature and should be used only by developers who have a strong grasp of the fundamentals of the language.

- **Dynamic Class Loading: -** java virtual machine (JVM) provides an important characteristic known as dynamic class loading. This has the ability to install the software at run time. Example is an Applet. Dynamic class loading provides the concept to written and testing the code to add new functionality without changing the previous code. That is it provides the facility to add new classes without having to recompile the whole thing.

3.4 Mobile Agents Attributes: -

MA is an entity that consists of five attributes [25],listed as follows:-



.

Figure 3.2: Agent property

3.4.1 State: - When an agent travels from one place to another, it transports with its state. So that it resume the execution at the destination node. The agent state is nothing but only the snapshot of the execution. In most programming language the agent's state is a combination of the execution state (or runtime state) and object state.

3.4.2 Implementation: - MA needs to code for their execution just other computer program. Scripting and interpreted languages both offers platform independence, which make them a viable solution for heterogeneous machine.

3.4.3 Interface: - An agent provides an interface to interact with other agent and other system. An interface can be anything that allows other agents and applications to access the methods on the agent for passing the message such as knowledge query and manipulation.

3.4.4 Identifier: - Each agent has a unique identifier during the lifetime of the agent from birth to death of the agent. An identifier is required; an MA can be identified and located for the directory services. For example, it can be a conjunction of the principals, identities and serial number.

4.4.5 Principals: - It is an entity that identifies an MA by any authenticated system. Principals can be individual, an organisation or corporation.

3.5 Agent behaviour: -

Transfer activity of the agent can be done itself by the agent or by another agent which is residing in the same place or by another agent which is residing outside the place. The agent is dispatched from the current place and received at the destination. The agent transfer process from the origin to destination point can be described as follows:-

3.5.1 Dispatching an Agent: - Preparing the trip, the MA must be identifying the destination. If the place is not defined, then agent is runs in a default place selected by the destination agent system. Once the destination is established, the MA informs the local agent system. When the agent system receives the agent's trip request, the following action taken:-

- **Suspend the agent: -** The agent is warned about the forthcoming transfer and is allowed to prepare for departure.
- **Serialize the agent: -** its state and the agent class are serialized by the engine. It is process of creating a persistent representation of the agent objects. It may include the execution state also.
- **Encode the serialized the agent: -** The engine now encodes the serialized agent for the chosen transport protocol.
- **Transfer the agent: -** The engine establishes the network connection to the defined destination host and transfers the encoded serialized agent.

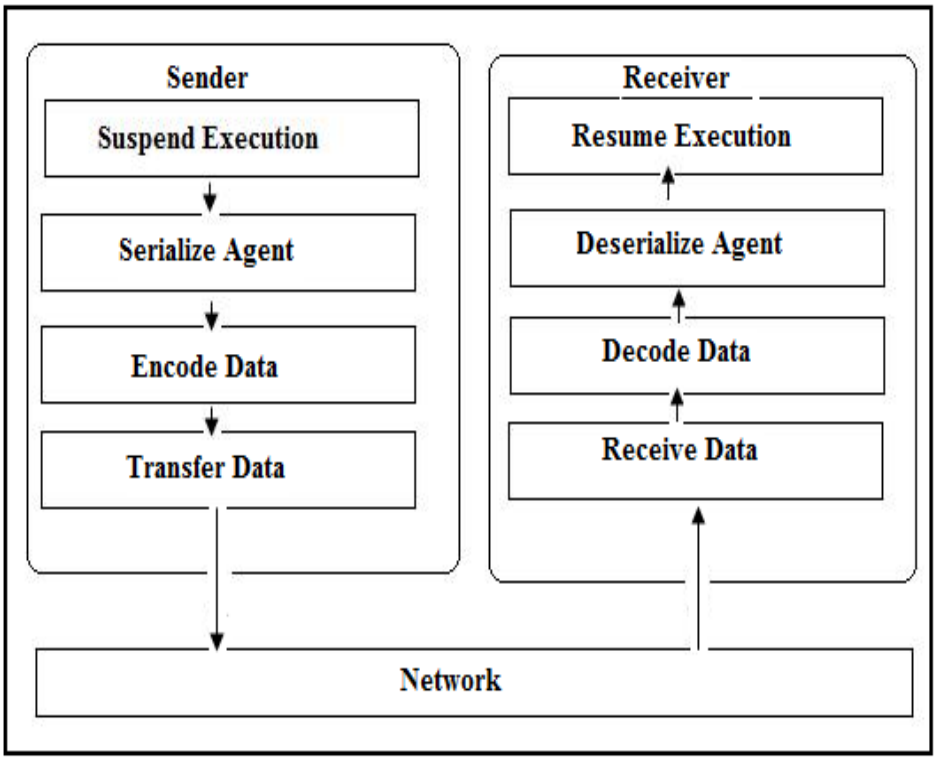


Figure 3.3 Agent transfer

3.5.2 Receiving an Agent: - Confirmation is done by engine when an agent is received. Only the senders who get authentication acknowledgement by the authentication manager will actual data transfer take place:

- **Receive the agent:** When agreement is done by engine to transfer the data, the encoded agent is received.
- **Decode the agent:** decodes the data stream.
- **Deserialize the agent:** the transferred agent state is restored.
- **Resume agent execution:** re-created agent is notified at the destination place. It can now prepare to resume it's the execution.

One stationary agent is required for all the agents, for the local hosts where agent is running. When an agent finishes its tasks, it submits its result to the local host and dies that's by results travel on the network. When an agent is received at a host, it may be cloned and forwarded to other active hosts, if required. One clone of the agent runs on the host at which cloning is done. After finishing their assigned tasks, agents submit results to their respective hosts. The clone executing hosts sends results of the cloned agents to the host at which the initial cloning was done. These are combined and sent back to the agent launcher.

3.6 Mobile Agent Itinerary: - Itinerary means simply the path which is followed by any agent (does not matter agent is human agent or software agent). Itinerary is the data structure which comprises many destinations. Each destination describes the location where the agent travels and does the assigned job. Itineraries of the system can be defined as the manner in which host are listed in it. Itineraries [28] can be determined by either statically or dynamically. Therefore it can be calculated either before or after the agent dispatch.

Itinerary may be static or dynamic [27] and then it can be classified as serial, parallel and hybrid itinerary. Best itinerary saves the parameter such as computation-efficiency, power-efficiency, and flexibility. The various itineraries are described in Figure 3.4 below:-

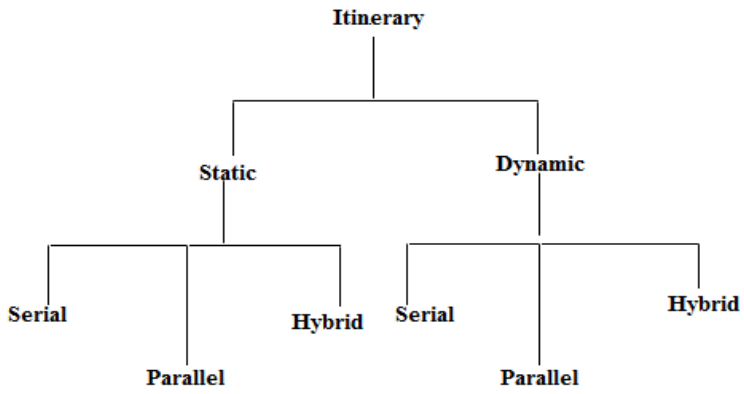


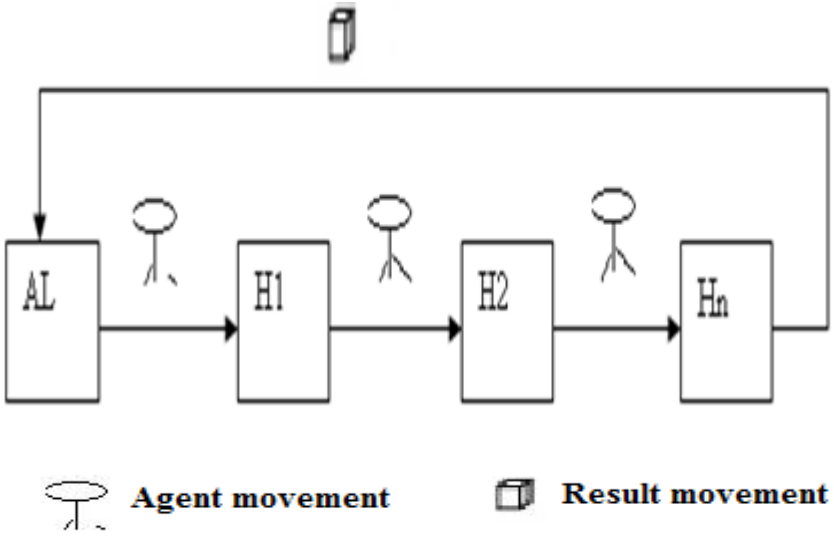
Figure 3.4: Itinerary of Mobile Agent

3.6.1 Static Itinerary: - Static means all things should be defined before doing the job. In reference of MA, all hosts that will be traversed during the job defined by client/user is specified before the launching the agent. When the agent moves from one host to other host it knows whole itinerary which is the initial host then be the next host. Static itinerary may be serially, parallel or hybrid. In this itinerary agent know the all parameter which is required for the pursuing the job, such as host id, host name, etc. It is simple but not able to environment adaptation. That's why a new host cannot introduce during the job.

3.6.2 Dynamic Itinerary: - Dynamic means it can able to adopt run time entity. In context of the mobile agent itinerary list can be modified at run time. It does not matter how many host's information in the list is available; but at least one host information should be in

the itinerary list so that agent can be launched. Agent launcher launches the one agent and then automatically that the next host information is determined. In dynamic itinerary, agent is so much intelligent as it can find the movement itself. Agent collects the all necessary parameters which will be required for travelling the agent. It may be serially, parallel, or hybrid. The implementation of dynamic itinerary is not too easy but it is very helpful for saving the network bandwidth.

- **Serial Itinerary: -** Serial means step by step. An agent travels as shown in Figure 3.5 from host to other host on the network with the required information. If any problem occurs in the itinerary such as any host in the list is currently unreachable then leave that host and go for next host and it may be tried again after visiting the next host in the list or it may be put at the end of the list in the itinerary and retried after all reachable hosts have been visited. This process is continued till the itinerary is exhausted or no more hosts are reachable. The agent then submits its result to the last host in the itinerary and dies. The result, however, is sent back to the agent launcher (client/user) by the last host in the itinerary.

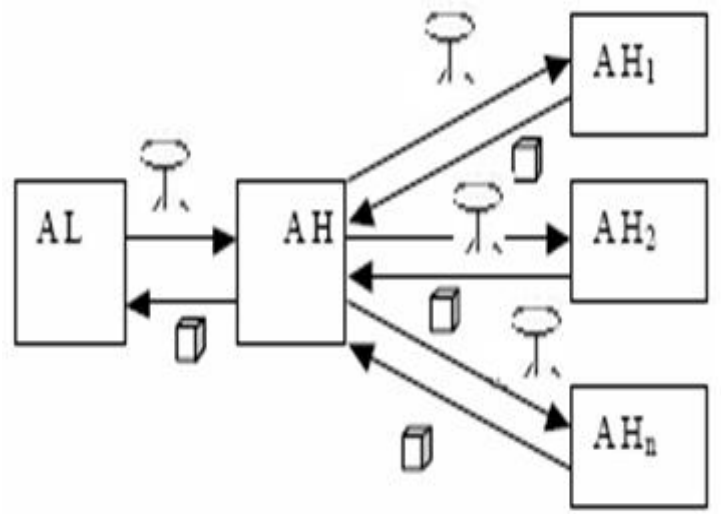


AL is agent launcher and (H1, H2, ...Hn) are hosts.

Figure 3.5: Serial Itinerary

- **Parallel Itinerary: -** In this itinerary, as shown in Figure3.6 clones are dispatched concurrently to all hosts in the networks. This type of situation is possible in network management system. If any host is unreachable due to any network problem or other problem, it is tried again.

Agent launcher creates the agent, if any host is unreachable, it is tried again. After dispatching all the worker agents, the master agent is ready to receive results from remote hosts. When all the results have been received, it dispatches this result to the agent launcher. The worker agents also have the facility to dispatch their own sub-worker agents, whenever and wherever essential, either in serial or in parallel manner.



AL is agent launcher and (H1, H2, ...Hn) are hosts.

Figure 3.6: Parallel Itinerary

When an intermediate child agent is required another child agent in the sub itinerary then prefers the virtual serial itinerary otherwise use the parallel itinerary, in order to save the time. On behalf of this, itinerary execution time can be optimised.

- **Hybrid Itinerary: -** Hybrid itinerary is the combination of serial and parallel itinerary. Sometime agent move in serially and sometime agent move in parallel depending on the condition. It can be either static or dynamic.

3.7 Mobile Agent life state log structure: -

The life cycle of MA begins at the moment when it is created. When MA migrates from one host to another host in order to achieve its goal; and the MA returns back to its server on which it was created. Two or more than two states, may be occurred at the different time or place in life cycle of MA. The mobile agent life state log structure [26] can be defined in four-tuple:-

Life_State_Log_Structure= (Agent_ld, State_Type, Time, Place),

Where,

- **Agent_ld:-** identifies a log item belongs to which mobile agent;
- **State_Type:-** indicates the type of mobile agent life state, with State_Type € STATUS={Creating, Running, Suspending, Migrating, Deleting }.
- **Time:**-indicates the time when the mobile agent (Agent_ld) came to the current State Type;
- **Place:-** identifies the agent server where the mobile agent (Agent_ld) came to the current State-Type at the specific time.

3.8 Life cycle: - The model of mobile agent paradigm is based on the migrating workflow [6, 7] system model. The resuming instance is the task executor in the migrating workflow system; it is a mobile agent in essence. The workflow-oriented life cycle model consists of five life states, (creating, running, deleting, suspending, resuming) and a number of transitions (active, suspend, dispatch, resume, terminate) between these states. The workflow-oriented life cycle model of mobile agent is shown in Figure. 3.7.

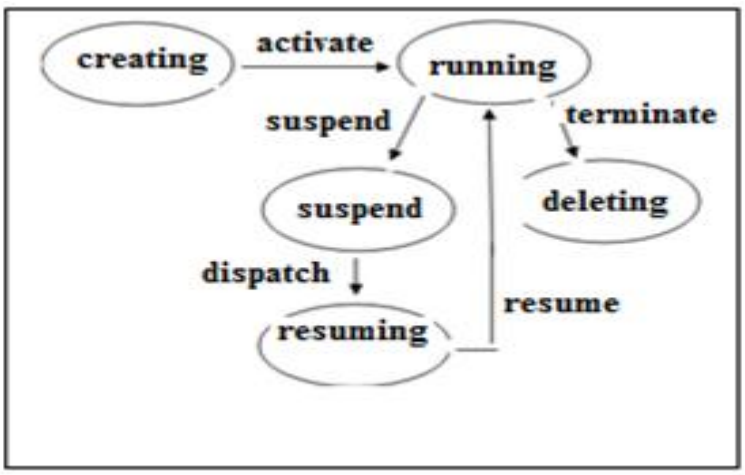


Figure 3.7: Life cycle of Mobile Agent [24]

To accomplish its task, the mobile agent can transport itself to another server in search of the needed resource/service, spawn new agents, or interact with other stationary agents. Upon completion, the mobile agent delivers the results to the sending client or to another server.

i. In the creating state, the agent is created but not activated yet.

ii. In the running state, the agent is running, performing actions and solve it pursue.

iii. In the deleting state, the agent is terminated;

iv. In the suspending state, the agent cannot run and still stay within the agent server;

v. In the resuming state, the agent is travelling between two server instances.

3.8.1 Life cycle phases :-MA adopts different phases throughout its life-time and can be listed as follows:-

i. Creation: - a brand new agent is born and its state is initialized.

ii. Dispatch: - an agent travels to a new host.

iii. Cloning: - a twin agent is born and the current state of the original is duplicated in the clone.

iv. Deactivation: - an agent is put to sleep and its state is stored on a disk of the host.

v. Activation: - a deactivated agent is brought back to life and its state is restored from disk.

vi. Retraction: - an agent is brought back from a remote host along with its state to the home machine.

vii. Disposal: - an agent is terminated and its state is lost forever.

3.9 Comparisons:- We have explained about the MA paradigms and its comparison with other computational paradigms, are listed in the table 3.1, below.

Table 3.1

Paradigms/Attributes	**Mobile Agent**	**Remote Evaluation**	**Client-server**
Implementation	Hard	Easy	Very easy
Security	Very low	Low	Very high
Performance	high	Very high	Low
Elements a) Data b) Code **c)** Stack	 semimobile mobile mobile	 static mobile static	 mobile static static
Itinerary	Static/Dynamic	Both	Static
Mobility	Code to data	Code to data	Data to code
Platform	Dependent	Dependent	Independent
Programming code	Hard	Hard	Easy
Examples	Aglet	Aglet	CORBA,

Comparison between various Distributed Computing Paradigms.

3.10 Agent oriented programming languages: - Before, we conclude this chapter; it is important to discuses some agent oriented programming languages. We will discuss AGLETS (java Agent Applets developed by IBM) and ODYSSEY (developed by General Magic's)

3.10.1 Aglet: - Aglet developed by a Japan based Company IBM. Some of their work done by the Object Management Group (OMG) [23] for consideration in regard to OMG's request for a Mobile Agent Facility (MAF). An Aglet is a mobile java objects that visits all aglet enables host in a network. It is autonomous because its run own thread of execution.

The key abstractions are aglet, proxy, context and identifier. For the network-based applications Aglets create a visual environment that uses mobile agents to search for, access and manage corporate data and other information.

In this section, we present briefly about “Aglets”, the platform permitting to handle and execute mobile agents which we used to implement our system of intrusion detection. Indeed, Aglets [21] (Agents Applet) are Java mobiles objects which can move from one machine to another. Thus, Aglet which is carried out on a host can stop its execution, to be off-set towards a distant host and continues this execution on the new environment. An Aglet API is a standard based on java, proposed to develop mobile agents. It was developed by a team of researchers of the research laboratory of IBM in Tokyo at the beginning of 1995. Its goal is to provide a uniform platform for the mobile agents in a heterogeneous environment such as Internet. An Aglet API defines the fundamental functionality of the mobile agents.

Aglets are autonomous because once you start them; they decide where they will go and what they will do. They can receive requests from external sources, but each individual aglet decides whether or not to comply with external requests. Also, aglets can decide to perform actions, such as travel across a network to a new computer, independent of any external request.

Aglets are a special type of Agents developed by IBM Research. There is an Aglet application programming Interface (API), and an Aglet Software Development Kit (ASDK) that is freely available on the Internet. Aglets are Java Objects that are able to travel from host to host (Mobile Agent). Aglets are hosted by an Aglet Server that provides the environment for them to run in. The Java Virtual Machine and a special Aglet security manager ensure safety while receiving and hosting Aglets. The word *Aglet* means “lightweight agent” it is composed of the word *agent* and the word *applet* (lightweight application).

Aglets workbench includes an Agent Web Launcher named Fiji and a Visual Agent Manager named Tahiti. Fiji is a Java applet based on the Aglets Framework and therefore capable of creating an aglet and retracting an existing aglet into a client’s web browser. Tahiti uses a unique graphical user interface to monitor and control aglets executing on a given computer. It also implements a configurable security manager that provides a fairly high degree of security for the hosting computer system and its owner. Aglets Workbench is a very versatile tool for creating secure mobile agent-based applications.

3.10.2 Odyssey: - First mobile agent implementation in pure java is Odyssey which is developed by General Magic's. The current Odyssey supports remote access of CORBA object and relational database via JDBC. It borrows from many of General Magic's concepts in the Telescript [23] tool set. Odyssey provides support for developing distributed mobile applications by implementing a set of Java class libraries. Odyssey technology implements the concepts of places and agents. It models a network of computers, however large, as a collection of places. A place offers a service to the mobile agents that enter it. A communicating application is modelled as a collection of agents. Each agent occupies a particular place. However, an agent can move from one place to another, thus occupying different places at different times. Agents are independent in that their procedures are performed concurrently. Odyssey is a set of Java classes and java threads for mobile agents and stationary places. They rely on the same security services as all other Java applications. The developer must adhere to a very rigid structure while implementing mobile applications on it. Some of the shortcomings of these platforms are stated as under:-

- They have minimal support for traditional distributed computing and treat agents differently than regular objects. Aglets uses sockets and Odyssey uses RMI to move agents between machines, but neither allows one to send a regular Java message to a stationary or moving agent. This limitation make it is very difficult to communicate with an agent after it is launched, and very hard for agents to communicate with other agents.

- Neither Odyssey nor Aglets allows one to create easily a remote agent, requiring one to build an agent locally. Aglets permits to send a string command to a stationary agent at a well-known URL, but the agent is responsible for decoding the command and executing it, which is a tiresome chore for a developer to program. Both platforms therefore make it very hard for objects to communicate with agents and for agents to collaborate across a network.

- Neither Odyssey nor Aglets allows an agent to move to an object, even if the object is stationary. In addition, neither platform permits a regular Java object to move.

- Odyssey and Aglets both include a special itinerary API that achieves the same behaviour but with more complexity to the developer. Neither Odyssey nor Aglets

allows agents to move between applets or from an applet to an application. Agents created with either of these platforms are thus significantly restricted in their ability to move.

- Odyssey and Aglets both include a simple local lookup table that allows one to associate a string with an agent's URL, but neither platform contains a distributed naming service that allows one to connect to a moving agent based on an alias. This limitation makes it difficult to locate a moving agent.

3.11 Limitations of the Mobile Agent Paradigm: - Security and robustness is the big issues of the mobile agent paradigm [34, 35]. A malicious mobile agent can damage the host. For example, is a Virus can be concealed as a mobile agent and distributed in the network causing damage to the host machines that execute the agent. On the other hand a malicious host can tamper with the functioning of the mobile agent. The agent can migrate from one place to other place automatically since it is too much intelligent same thing happen with the malicious agents, similar to viruses; expose these hosts to the risk of system penetration. Unless some countermeasures are taken, such agents can potentially leak or destroy sensitive data and disrupt the normal functioning of the host.

Buggy agents can use consumption of resources such as CPU time, disk space, and network connections or even screen space, thereby denying their use to other agents and legitimate users of the host. Security mechanisms are thus necessary to safeguard hosts' resources from the agents executing on them.

Although the Java language system is highly suitable for creating mobile agents, but there are some significant shortcomings. Some of them can be worked around; others are more serious and have implications for the overall conceptual design and deployment of Java-based mobile agent systems. Some of the Java shortcomings [3, 6] are inadequate support for resource control, no protection of references, no object ownership of references, no support for preservation and resumption of the execution state.

3.12 Applications of mobile Agents: - There are wide areas of mobile agent applications [33]. Agent applications are developed for different fields such- as the Information Retrieval Systems (IRS), Distributed File System, Clinical Data analysis for medical diagnosis, Distributed Data Mining, Distributed Real-time systems, Mobile Wireless

Environment, Mobile Smart Databases, Peer-to-Peer Computing, Network monitoring and management, Intrusion Detection System, Network routing, Performing location-dependent computations, Load balancing, Service customization, Wireless Sensor Networks(WSN)/ Remote Sensing, Wireless Ad hoc Network (WAHN), Manufacturing, Command & Control, Grid Computing/Cluster Computing, and Information dissemination etc.

Information search and altering applications is the most important feature of the mobile agent, they can execute on the server machines and access the data without using network. After the execution on the server it sends back only results to the end user. Thus the bandwidth consumption will be less. While in traditional application they can execute on server machine, execute the data and get very small result data. That's why traditional approach uses more bandwidth.

In a web based application; there are often several network connections for each application. If this job done by traditional computing paradigms then it used lot of network bandwidth to maintaining the network connections. Instead of traditional computing paradigms, mobile agents are used, the client does not have to maintain a network connection while its agent's access and process information. This permits increased asynchrony between the client and server. This feature is especially useful for mobile computers such as laptops and personal digital assistants (PDAs), which typically have low bandwidth, unreliable connections to the network and are often switched off to save power consumption. Also, the repeated client/server interactions are reduced to agent transfer operations, thus reducing the frequency of network usage as well.

The mobile agent paradigm can be exploited in a variety of ways, ranging from low level system administration tasks to middle-ware to user-level applications. If the application uses remote procedure calls to control a device, it may be difficult (if not impossible) to guarantee that it will meet the real-time (system level application) deadlines associated with the device. This is because communication delays are usually neither bounded nor accurately predictable, unless the underlying networks provide quality of service guarantees. Instead, the application can send an agent to the device and control the device locally, resulting in better predictability. Other system level applications are network maintenance, testing and fault diagnosis, installing and upgrading software on remote machines, etc.

3.12 Summary: - MAs are a promising solution for design and implementation of large scale distributed applications, as they overcome many drawbacks of the traditional client server approach. This is a clear evolutionary path that will take us from current technology to widespread use of mobile code and agents within the next few years. Once several technical challenges have been met, use of mobile agents will expand rapidly.

CHAPTER 4

PMADE BASED PROPOSED MODEL

As already discussed in section 1.2 (page 2) of the thesis, regarding the concept on Data gathering (DG) or data mining (DM). We now elaborate upon this concept as it forms the basis of our work. The three main constituents, Data, Information and Knowledge, can be summarized as follows: -

i. **Data: -** Data are any facts, numbers, or text that can be processed by a computer. Today, organizations are accumulating huge and growing amounts of data in different formats and different databases. This includes:

   - operational or transactional data such as, sales, cost, inventory, payroll, and accounting
   - nonoperational data, such as industry sales, forecast data, and macro economic data
   - meta data - data about the data itself, such as logical database design or data dictionary definitions

ii. **Information: -** The patterns, associations, or relationships among all this data can provide information. For example, analysis of retail point of sale transaction data can yield information on which products are selling and when.

iii. **Knowledge: -** Information can be converted into knowledge about historical patterns and future trends. For example, summary information on retail supermarket sales can be analyzed in light of promotional efforts to provide knowledge of consumer buying behavior. Thus, a manufacturer or retailer could determine which items are most vulnerable to promotional efforts.

The last decades have seen an explosive growth of stored data due to advances in automated data collection tools such as sophisticated telescope, remote satellite sensors, global positioning systems, etc. Major sources of stored data in business include web, e-commerce, retail transactions, stocks, etc.; in science include telecommunications, geo-sciences, astronomy, meteorology, bioinformatics, scientific simulation in chemical engineering, fluid dynamics, etc. News archives, digital camera images also create large data sets. We are rich in data, but still striving for the knowledge.

Exponential growth in data stored in large data repositories had made the human being (decision makers) helpless in getting information without powerful tools. Data mining (or knowledge discovery from data) is automated analysis (extraction) of interesting data patterns representing knowledge implicitly stored in large databases, centralized data warehouses, the Web, other massive data repositories, or data streams. Data Patterns are implicit, previously unknown and potentially useful.

Some alternative names for Data Mining are Knowledge discovery (mining) in databases, knowledge extraction, data/pattern analysis, data archaeology, data dredging, information harvesting, business intelligence, etc. Data mining helps in turning the data tombs into "golden nuggets" of knowledge.

4.1 Distributed Data Mining: - Most of the existing DM techniques were originally developed for centralized data and need to be modified for handling the distributed case. The increasing demand to scale up to massive data sets inherently distributed over a network with limited bandwidth and computational resources available motivated the development of methods for parallel knowledge discovery (PKD) and distributed knowledge discovery (DKD). The related pattern extraction problem in DKD is known as Distributed data mining (DDM).

In a distributed computing environment like internet, intranet, LANs, wireless networks, sensor networks, etc., Distributed applications uses Data Mining techniques to analyze and monitor large data sources. Distributed Data Mining (DDM) aims at extraction useful pattern from distributed heterogeneous data bases in order, for example, to compose them within a distributed knowledge base and use for the purposes of decision making.

Distributed data mining requires an architecture which is completely different from the one used in centralized approaches. In a distributed environment, the architecture must facilitate to pay careful attention to distributed resources of data, computing, and communication

There are so many systems that have been developed to help users as they work, by performing automatic Web search for information to support tasks such as Web browsing, query generation, and document authoring, by mining the Web and other resources. Reflecting context has been recognized as important to realizing the potential of Web search in general, and context-sensitivity plays an especially crucial role in proactive retrieval systems. The degree to which the system can provide context-relevant information determines whether the system will be an aid or an annoyance. Unfortunately, full

exploitation of contextual information during Web search is challenging. The current search engines, there are strong limits on query length (e.g., Google's query length limit of ten terms), making it difficult to provide enough terms to describe rich contexts. Even if an adequate context description can be included within the limits, there is no guarantee that the vocabulary used to describe the context will match the vocabulary by which the resource is indexed.

Context searching is one of the important searching, it is based on the end-user can select which tags (patterns) or elements (patterns) within a document to search on within one or more documents. Element indexes documents according to individual in a document, rather than just to the document level as most search engines do. Element provides an intuitive interface that allows users to easily select what contexts they wish to search, without having to know the structure of the document nor complex Path syntax. Element technology can also highlight search terms and link directly to the point in the document in which the "hit" is located, when searching and displaying document. This allows common users to find the information they need, when they need it, within just a few clicks.

This thesis presents data gathering using Mobile Agent and we have used the PMADE platform (Platform for Mobile Agent Distribution and Execution)[39]. Data gathering is also known as DATA MINING. It is expected to perform partial analysis of data at individual sites and then to send the outcome as partial result to other sites where it is sometimes required to be aggregated to the global result [14]. The migration of MA can be classified into two types: -

- **Strong Mobility:** The system to support strong mobility if the execution state is migrates along with the unit of computation. In systems supporting strong mobility, migration is completely transparent to the migrated code. There are many platforms such as NOMADS, Ara and D'Agents which based on strong mobility. In strong mobility, the entire state of the agent, including its execution stack and program counter be saved before the agent is migrated to its new destination. It is a complicated task to realize, and typical implementations of this functionality in MA systems.

- **Weak Mobility**: On other hand the systems that discard the execution state across migration are said to be *weak mobilit*y. With weak mobility, extra programming is required in order to save part of the execution state. Many of the mobile agent systems support weak mobility viz., PMADE (Platform for Mobile Agents Distribution and Execution), aglets, and Ajanta.

Many of the mobile agent systems support weak mobility viz., PMADE, Ajanta and Aglets. Most of the agent systems are implemented on top of the Java Virtual Machine (JVM), which provides with object serialization basic mechanisms to implement weak mobility. The JVM does not provide mechanisms to deal with the execution state. Security is of great importance in mobile agent systems, not only in electronic monetary transactions, but also those mobile agents should not become the next generation of viruses. Current research on secure agent systems concentrates mainly on protecting hosts against hostile mobile agents. The problem of security stems from the fact that untrusted code needs to be executed. Modern solutions are based on the notion of protection domains by which a security policy for accessing local resources can be enforced. Only very few systems also provide facilities for protecting mobile agents against hosts, but PMADE is the platform which takes care all round security, i.e., Agent-Agent, Agent-Host, Host-Agent and Host-Host.

4.2 PMADE (Platform for Mobile Agent Distribution and Execution): -

Peer to Peer (P2P) networks can be used to tie together the computing power of hosts in a network. In multi-agent systems, the basic interaction pattern between agents is P2P. For the diverse connectivity, a very popular concept P2P is used. It is save the bandwidth of network participants rather than conventional centralized resources where a relatively low number of servers provide the core value to a service or application. For connecting the nodes via largely *ad hoc* connections P2P networks are used. Such networks that are P2P are very useful for many purposes such as sharing the files containing audio, video, data or anything in digital format is very common, and real time data, such as telephony traffic, is also passed using P2P technology. P2P concept is used in PMADE framework. The migration of MAs is associated with different movement costs, viz., transmission time, round trip time, number of hops, etc. Costs are also generated by executing the agent's algorithms on the agent host (AH) [14]. Autonomous agents can observe their environment and reason and act on the basis of these observations.

The integration of the functionality of P2P within the **PMADE** framework for large-scale multi-agent systems (MASs) is to simplify P2P application and service development and deployment by freeing the application developer of all low-level details including communication, security and scheduling. Large-scale MASs [13] needs to satisfy the multiple strict requirements such as reliability, security, interoperability, scalability, trustability,

reusability and maintainability.

The important characteristic of PMADE is: -

(i) To provide a platform for large-scale agent systems

(ii) Support multiple code bases and operating systems

(iii) Interoperability with other agent platforms

(iv) Reduce network traffic

PMADE is a new platform. It provides a very simple, yet effective interface for the common user. It is client-Server based paradigm but difference is it uses the code to data approach for information gathering. Therefore it can save the bandwidth and resources as for needed. Each node of the network has a server called Agent Host (AH), which accepts and executes incoming agents and a client called Agent Submitter (AS), which is responsible for submitting the Mobile Agent on behalf of the user to the Agent Host.

Looking into the working of this model, how the Agent Host and Agent Submitter communicate to each other. When a user wants to perform a task or if the applicant requires gathering the information through a Mobile Agent, then the user submits the Mobile Agent designed to perform that task to the Agent Submitter. Agent Submitter checks the connection. If connection is ok with the specified Agent Host, where the user already holds an account; then the Agent Submitter submits the Mobile Agent to it and gets an acknowledgment and can then go offline. If connection is not established with the Agent Host then it pass appropriate message to user. PMADE has two main components as shown in Figure 4.1.

4.2.1 Agent Submitter: - Agent Submitter is the intermediate or interface for the user and Agent Host as shown in Figure 4.1. One of its primary jobs is to attach a signature to the newly prepare Mobile Agent. It retrieves the static IP address of the Agent Host and binds it to the Mobile Agent signature. This enables the Mobile Agent at the secondary Agent Host to send the reply back to the parent Agent Host.

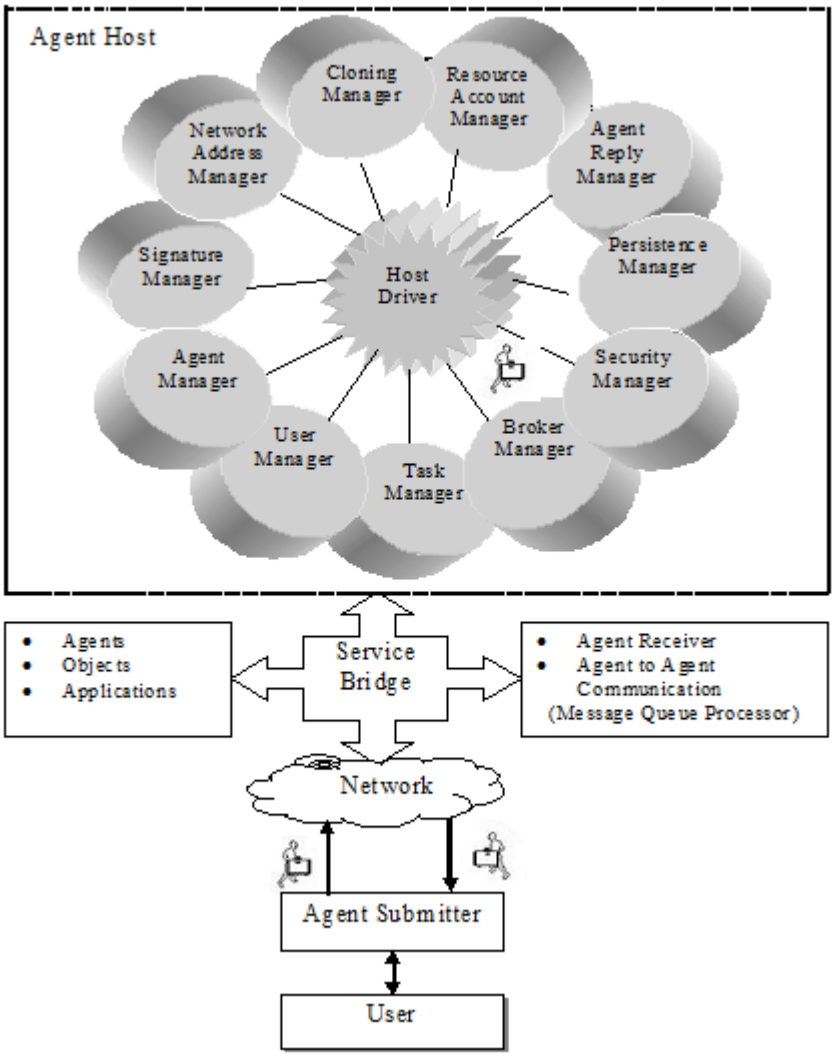


Figure 4.1: Detailed architecture of PMADE model [39]

4.2.2 Agent Host: - The key component is Agent Host in the PMADE. It mainly consists of two modules manager modules and the Host Driver. The Host Driver is lies below the managers modules and is responsible for driving the AH, it provides the proper co-ordination between the various managers. The Service Bridge provides hosting facility to a carrier agent. The carrier agent regularly monitors the requests issued by the AH. If it receives a request to transfer an agent to another AH, it takes immediate action. It exploits the services of HTTP protocol for transferring an agent. Various manager modules help to perform various functions like- transfer, execution, and communication etc. of mobile agents. The following sections provide detailed descriptions of the managers of the AH in chronological order [39].

- **Task Manager (TM): -** The Task Manager is responsible for creating and maintaining a multi-level queue, depending upon the nature of the received agent. The job of this manager is to dispatch a particular agent for processing, when it deems fit. It also reserves the power to terminate and pre-empt the resources held by a Mobile Agent.

- **User Manager (UM): -** This module deals with the identification and verification of users who use this Agent Host as their Base Host.

- **Agent Manager (AM): -** The Agent Manager accepts Mobile Agents on behalf of the Agent Host. It checks whether it can accept any and if it can, it does, otherwise it sends a suitable reply to the respective Agent Submitter.

- **Signature Manager (SM): -** This manager does the bookkeeping for the Agent Host and maintains records of signatures of all the agents that have been accepted by the Agent Host for processing. This also addresses the problem of a malicious agent consuming the Agent Host services for infinite time.

- **Network Address Manager (NAM): -** This module keeps track of the network addresses of all the servers on which the Agent Host services are running and with which it is supposed to communicate directly for the purpose of forwarding the cloned agents.

- **Cloning Manager (CM): -** This module is responsible for making clones of all the Mobile Agents that are received by the Agent Host.

- **Resource and Accounting Manager (RAM): -** This is one of the most crucial modules of the architecture, if it has to be made economically viable. As its name suggests, this module is responsible for keeping a table on the resources used by a particular Mobile Agent, which uses the services of the Agent Host. It calculates the cost of resources used by the Mobile Agent, on the basis of parameters such as processor time consumed, bandwidth utilized, amount of secondary storage used by the Mobile Agent, etc. The parameters can be modeled to suit the specific needs of the consumer and the service provider.

- **Agent Reply Manager (ARM): -** This module is assigned the job of receiving and sending replies. It keeps on polling to establish the connection with those Base Hosts to which the results are to be sent.

- **Persistence Manager (PM): -** If the Agent Host has to be shutdown for some reason, it is the Persistence Manager who saves the state of all the Mobile Agents residing on the Agent Host. This allows the system to go offline for regular maintenance and yet not lose the Mobile Agents that are running on the Agent Host.

- **Broker Manager: -** This manager is in the process of designing. Brokering is a major requirement in e-commerce applications and can benefit vastly by the use of mobile agents. There may be various static agents & applications running on several servers and capable of providing different services at a cost. A Broker Manager can be implemented to guide an incoming agent to the appropriate static agent on the node. It maintains a record of all these service agents, the services offered and their charges, etc. It may also advertise these to the Broker Managers on other nodes. Whenever a mobile agent arrives on an AH, it would announce its requirements to this Broker Manager, who would then guide it appropriately.

4.3 The PMADE Framework: -

The proposed paradigm (Mobile agent) provides many advantages over trip time, network load and security etc. Firstly, only one stationary agent is sufficient for all the agents running on the local host. Secondly, when an agent finishes its assigned task, it submits its result to the local host in secure form and dies. Thus, only the results travel on the network. When the agent reaches a host and analyzes the network, if required to clone the agent then it cloned and forwarded to other active hosts, clones of the clone are further created by the agent itself. One clone of the agent runs on the host at which cloning is done. After finishing their assigned tasks, agents submit results to their respective hosts. The clone executing hosts sends results of the cloned agents to the host at which the initial cloning was done. These are combined and sent back to the agent launcher.

In Figure 4.2, three cloning stages, H_A, H_B and H_C are shown. The agent is received at stage H_A which is launch by the agent launcher; Host H_A clones are generated for next host and forwarded to hosts at stage H_B. Further, cloning is done by the agent itself and forwarded to stage H_C and so on. This cloning can continue on further stages, if required.

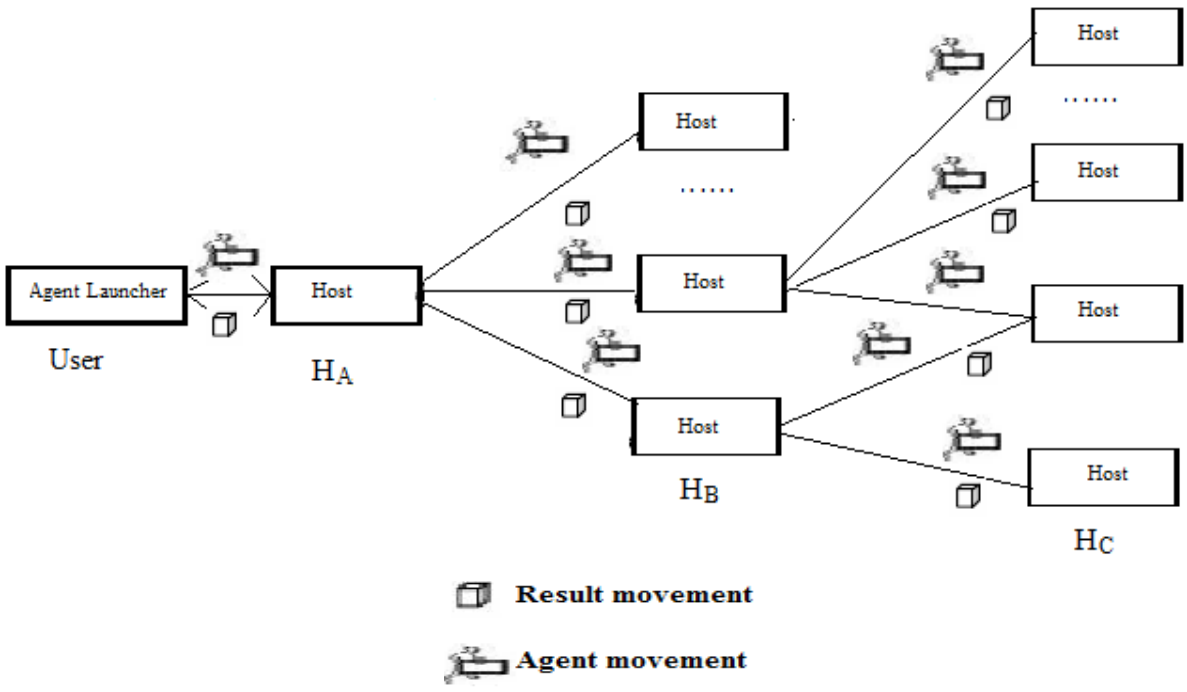


Figure 4.2: Mobile Agent in parallel and distributed computing

Now assume an agent $MA_1$of size S_a KB is received at stage H_A, m clones are forwarded to stages H_B and n clones to stage H_C. Assume that (B_1, B_2, B_3,.....B_m) *and* (C_1, C_2, C_3,.....C_n) KB of result data is generated by the cloned agents at stage H_B and H_C respectively, and stage H_A generates a result of AR KB. In the previous paradigm, the total amount of data that travels on the network after completion of tasks at different stages and before delivery of result to Agent Launcher (AL), is

$$D_o = S_a + AR + mS_a + \sum_{i=1}^{m} B + nS_a + \sum_{i=1}^{n} C$$
$$= Sa\ (\ 1 + m\ + n\ ) + AR + \sum_{i=1}^{m} B + \sum_{i=1}^{n} C$$

Where as in the new paradigm we have

$$D_n = AR + \sum_{i=1}^{m} B + \sum_{i=1}^{n} C$$

The design of PMADE includes the design of agents, objects and services, interactions, migrations, security and authorization, as well as the agent platform itself. For example, scalability of agents and objects is realized by distributing objects according to a peer object distribution strategy, but not agents. Each node of the network has an Agent Host (AH), which is responsible for accepting and

executing incoming autonomous Java agents and an Agent Submitter (AS), which submits the MA on behalf of the user to the AH. A user, who wants to perform a task, submits the MA designed to perform that task, to the AS on the user system. The AS then tries to establish a connection with the specified AH, where the user already holds an account. If the connection is established, the AS submits the MA to it and then goes offline. The AH examines the nature of the received agent and executes it. The execution of the agent depends on its nature and state. The agent can be transferred from one AH to another whenever required. On completion of execution, the agent submits its results to the AH, which in turn stores the results until the remote AS retrieves them for the user. The AH is the key component of PMADE. It consists of the manager modules and the Host Driver. The Host Driver lies at the base of the PMADE architecture and the manager modules reside above it. It is the basic utility module responsible for driving the AH by ensuring proper co-ordination between various managers and making them work in tandem. PMADE provides weak mobility to its agents and allows one-hop, two-hop and multi-hop agents.

Agent-agent interaction is exclusively via message-passing communication. Asynchronous message passing has good scalability characteristics with a minimum of synchronization between the agents. Agent migration between locations is based on weak mobility. The *state* of the agent is captured (e.g., the variables referenced by the agent) but not the context of the agent (e.g., stack pointer and program counter).

Agent host supports agents, objects servers and objects. It provides the interface and access to the agents that are hosted by the agent host. Service access providers make services accessible in PMADE. A location is a closely related collection of agent host and object servers, possibly on the same (high-speed) network, on hosts which are managed in the same administrative domain or different network domain. Each host runs a *minimal* AS or AH, and zero or more objects servers, and service access providers. The distributed AHs, the object servers, and service access providers implement a location. Depending on the policy or resource requirements, one agent can be exclusively assigned to one AH, or a pool of agents can be hosted by one AH. The explicit use of AHs makes some aspects in the lifecycle model of agents more clear. An active agent is assigned to, and runs on an AH; a suspended agent is not assigned to an AH. In this model, starting a newly created, or activating an existing suspended agent, is similar, and some design decisions of the agent lifecycle can be simplified. The use of agent and object servers is transparent to the agents. However, e.g., for performance or security

management reasons, an agent could ask the middleware to determine on which agent host the agent runs.

4.4 System Architecture :- Searching is the pattern based or text based search which allows to the end-user; and the end user select which tags to search in database. This will match the relevant data; return only relevant and less results. Return result is depends on the pattern or text, in many cases, much less results because they are more relevant. Gathering information will be faster as much as possible. The searching using the MA is much faster because it should not have to shift through thousands of results. For providing reliable solution we have developed a multi-agent system using MAs as shown in Figure 4.3. This system is divided into four sections [14], interface, agents, policy and Agent-Agent Communication Layers. The interface is used for the communication with external world via PMADE as well as for agent-agent communication. It maintains a buffer. The buffer is used for storing the messages temporarily whenever communication delay arises. It also helps to provide fault tolerance to the message on system failure. For providing the security to messages it uses PMADE security. We have considered few assumptions and identified some policies & agents in the development of the system.

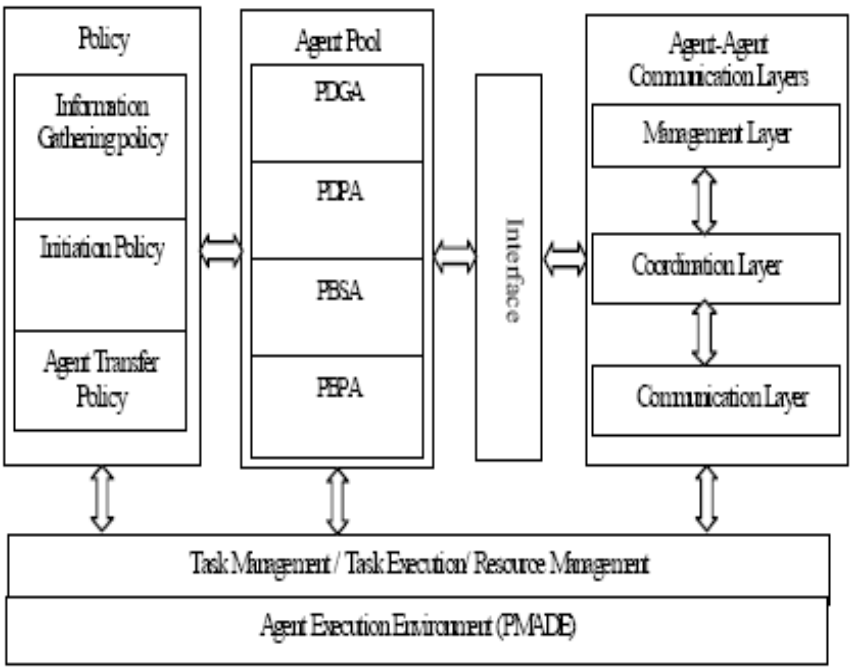


Figure 4.3: Mobile Agent based Distributed data gathering system [14]

4.4.1 POLICY: - According to need of the agents we have defined four policies. These policies are maintained by system administrator and updated according to need of searching schema.

- Information gathering policy
- Initiation policy
- Agent transfer policy and
- Server selection policy

All policies are governed by system administrator and updated according to the need of searching schemes.

4.4.2 Information Gathering Policy (IGP): - The strategy of searching the information including the all method and frequency of information gathering is defined in the Information Gathering Policy (IGP). The frequency is determined based on a swapping between the accuracy of load information and the overhead of information collection.

4.4.3 Initiation policy: - Who starts the searching process, determined by Initiation Policy (PI)? The process (MA) can be initiated by a client and it is monitored by intermediate servers. PMADE activated servers are being in-charge to process coming agent on a server as they are arriving.

4.4.4 Server selection policy: - Server Selection Policy (SSP) selects an appropriate server based on the load information, i.e., density of MAs to which the workload on an overloaded server can be reallocated. Different strategies can be applied to the selection. For example, the find-best strategy selects the least loaded server among all servers and this strategy selects the first server whose load is below a threshold. The least loaded server has been taken into best category as it has very less load and can be selected as an appropriate server for processing the agent and responding.

4.5 Mobile Agents: In our MA based data gathering system, there are four agents shown in Figure 4.4; on the behalf of these agents, we gathering the information from one host to other host. Out of four agents, two are MAs and other two are stationary intelligent agents. Detailed relationship among these agents is illustrated in Figure 4.3. These agents are:

- Pattern Database Generator Agent (PDGA),
- Pattern Database Processor Agent (PDPA),
- Pattern Based Search Agent (PBSA) and
- Pattern Based Processor Agent (PBPA).

Pattern Database Generator Agent (PDGA) and Pattern Based Search Agent (PBSA) is the MAs while *Pattern Database Processor Agent (PDPA) and Pattern Based Processor Agents (PBPA)* is stationary agent. The relationship among these agents is shown in Figure 4.4. These agents maintains static as well as dynamic itinerary. In the static itinerary all parameter defined before the launching the agent i.e. agent know everything about the host while in the dynamic itinerary agent is able to know the information about the host; that is whenever required agent can be updated at any node at any time in the itinerary. These agents maintain two containers (Result Container and State Container) one for transporting result data across the network and other for state variables and their intermediate values.

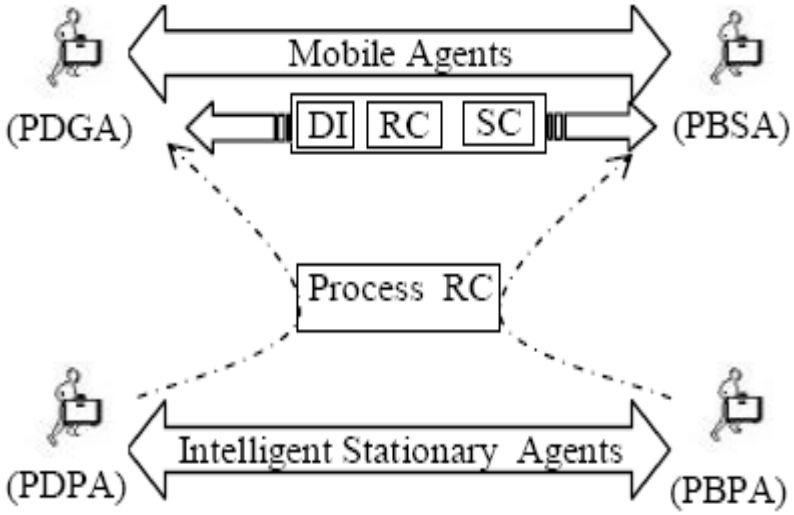


Figure 4.4: MA in data gathering [14]

The PDPA and PBPA process the Result Container of PDGA and PBSA, respectively. A brief introduction of these agent are as follows:-

File1 ⇒ *{F1,Ff2 …Fn}* – A set of files which is stored in the Public Store (PS) at each node.
File2⇒ *{F1, F2…Fn}* File2 ⊂ File1, – A set of Files containing the pattern, which is stored in the pdga-pdpa directory. These agents generate three document vectors [23]. These documents are:-

- **Pattern vector: -** Pattern vector (PV) contains all the different patterns or words in a file that are distinguished by the blanks in between the words.

- **Frequency vector:** -Frequency vector (FV) which contains the corresponding frequency of each pattern or word stored in the PV that is how many times that pattern exists in a file.

- **Location vector:** - Location vector (LV) that contains all the corresponding locations of the words stored in the pattern vector, i.e., all the locations at which that pattern or word exists in a file.

DI denoted to Dynamic Itinerary. *PS* denoted to Public Store containing F1**.** *PD* denoted to Patten Database containing the document vectors (File Name, PV, FV, LV) for each file containing the Pattern. *SC* **-** State container containing the state variables and intermediate values. *RC* **–** Corresponding Result Container of the mobile agents PDGA, and PBSA to be transferred back to AS.

4.5.1 Pattern Database Generator Agent (PDGA): -

In our system consist from four agents; first is PDGA. PDGA is dynamic and travels from one host to other host during the itinerary. It works on the behalf of the user. User or client gives the pattern to the agent and then agent starts its journey Figure 4.5. According to the pattern, agent searches all the files in Public Store (PS) at a node in which that pattern exists in the itinerary. After receiving all the files, three document vectors are generated by the PDGA for each file. There are pattern vector, frequency vector and location vector. P*attern vector (PV),* which contains all the different patterns or words in a file that are distinguished by the blanks in between the words. The second document vector is called the *frequency vector (FV)*, which contains the corresponding frequency of each pattern or word stored in the pattern vector that is how many times that pattern exists in a file. The third document vector is called the *location vector (LV)* that contains all the corresponding locations of the words stored in the pattern vector, i.e., all the locations at which that pattern or word exists in a file. It then creates Pattern Database (PD) at that node and stores File Name, PV, FV, and LV in it.

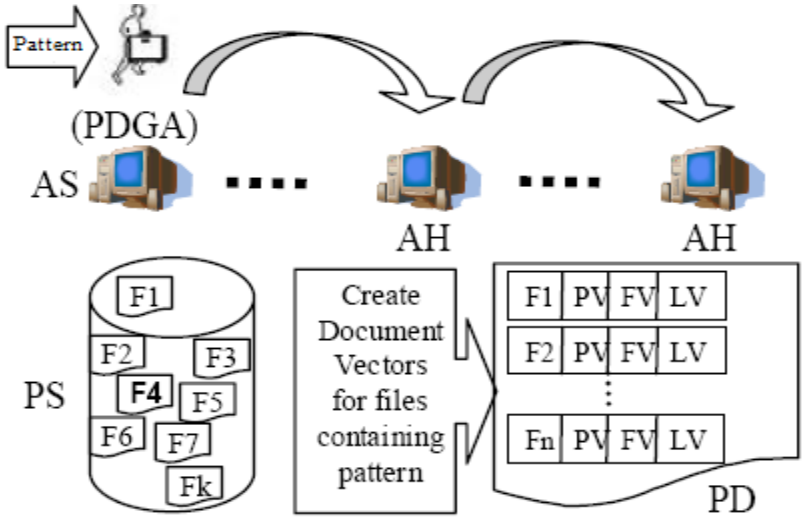


Figure 4.5: Working of PDGA [14]

Algorithm: - Pattern Database Generator Agent

Input: -A set of words or patterns

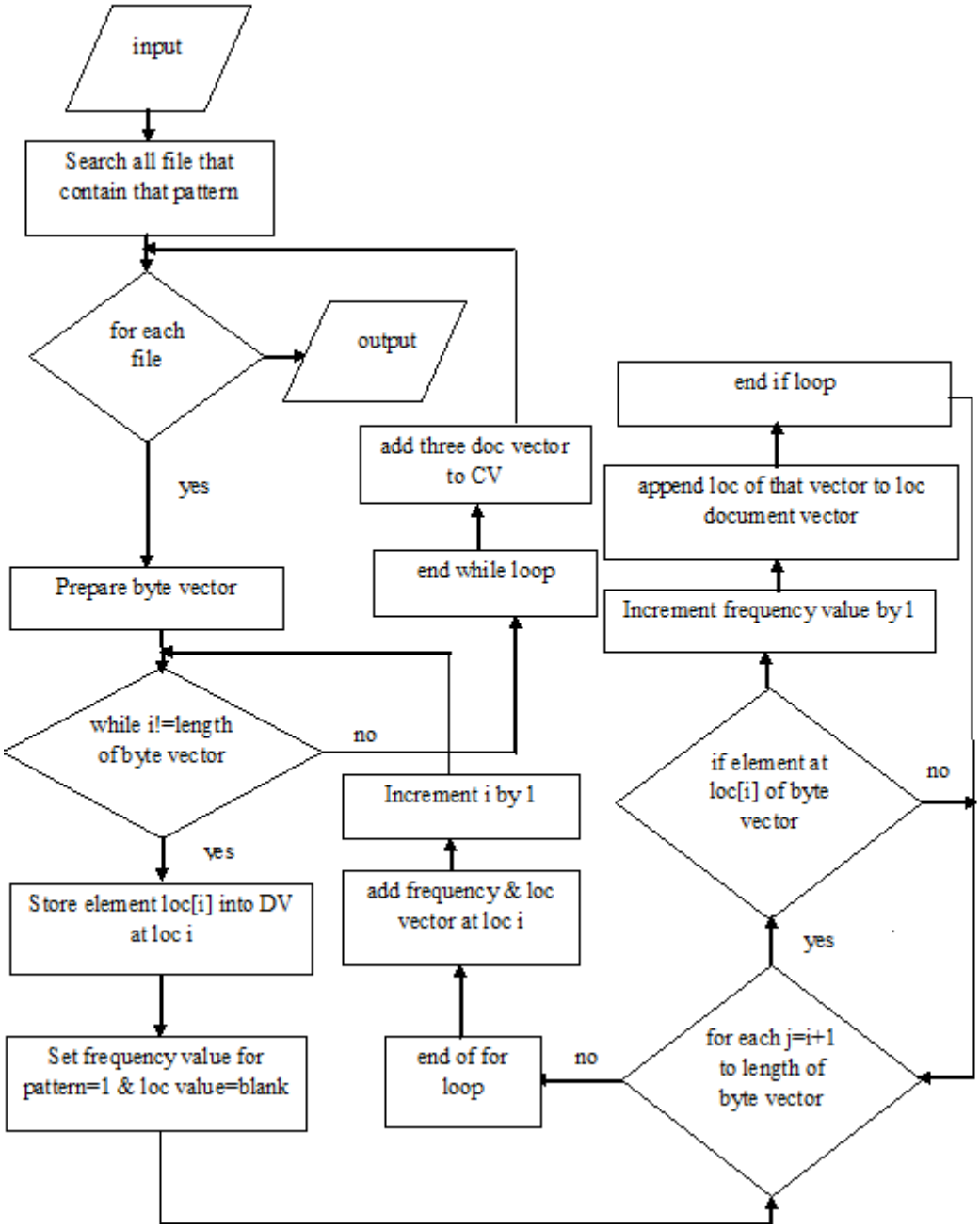


Figure 4.6 :Flow Chart of PDGA

Output: - A container that contains three document vectors: the pattern document vector, the frequency document vector and the locations document vector for each file that contains the pattern given by the client.

4.5.2 Pattern Based Search Agent (PBSA): - When a user or client gave an input (i.e. pattern or word) to the agent (PBSA); after getting the input agent starts its journey. We have discuses above three type of document have been generated by the PDGA. PBSA searches all the PV in PD on a node in which that pattern exists in the itinerary. This agent puts results (File Name, PV, FV and LV) on a node into the RC. Now PBSA is ready to migrate to next destination in the itinerary. When agent visits its final destination it submits Result Container to Agent Reply Manager (ARM) of the current visiting host and finally ARM sends result to the client (AS) of the agent.

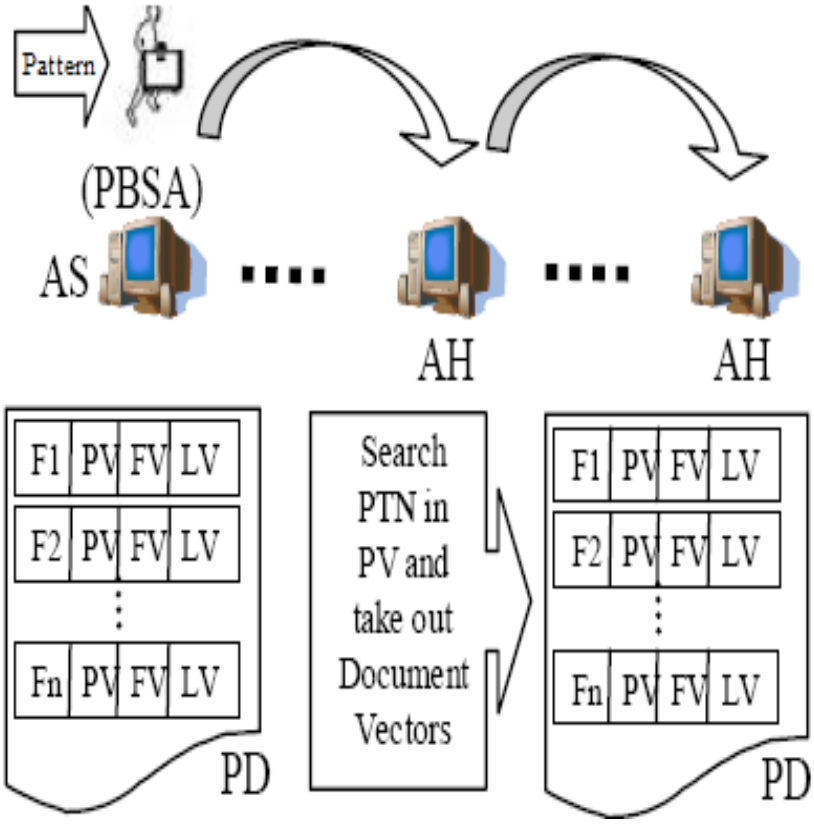


Figure 4.7: Working with PBSA [14]

Algorithm: - Pattern Based Search Agent

Input: -A set of word or pattern

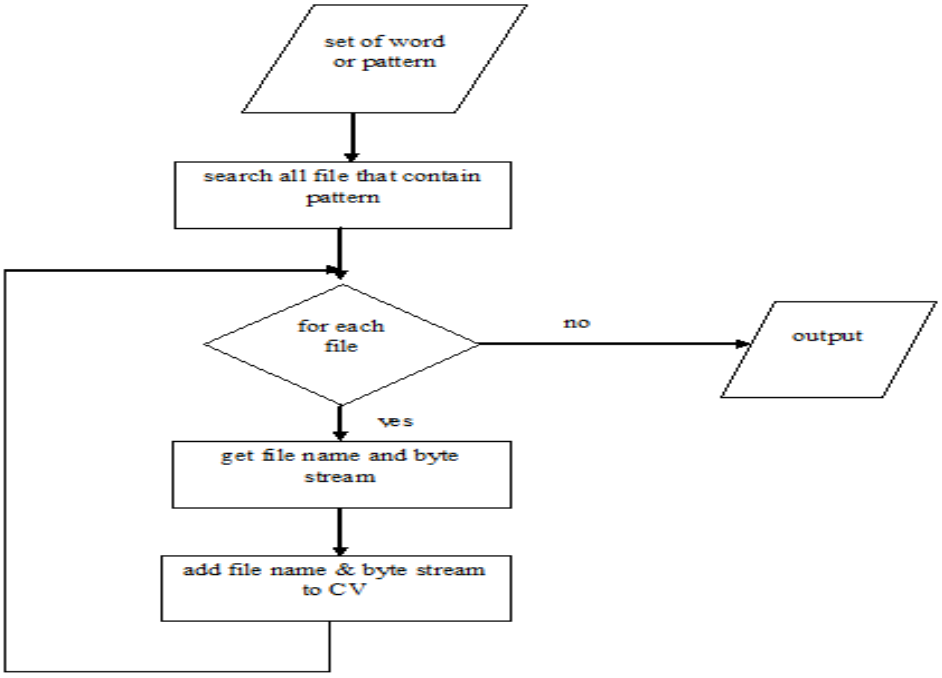


Figure 4.8 :Flow Chart of PBSA

Output: - A container that contains name of the file and their corresponding byte stream for each file that contains the pattern given by the client.

4.5.3 Pattern Database Processor Agent (PDPA): -

processes the Result Container prepared by the PDGA during the itinerary. It retrieves three vectors from the Result Container (*document vectors*¸ *pattern vector* and *frequency vector*) for every visited node in the itinerary. By processing these vectors it regenerates all the original files and stores at local client node.

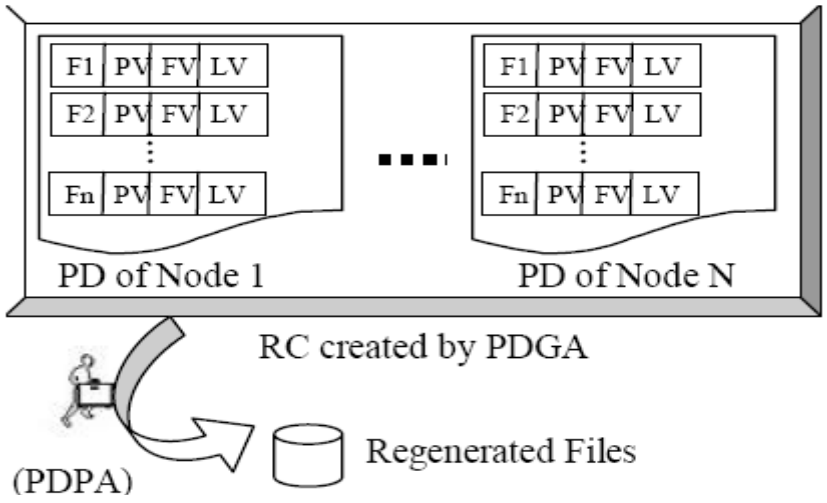


Figure 4.9: Working of PDPA [14]

Algorithm: - Pattern Database Processor Agent

Input: - A container vector that contains the pattern document vector, a frequency vector and a locations vector along with the name of the file for each file.

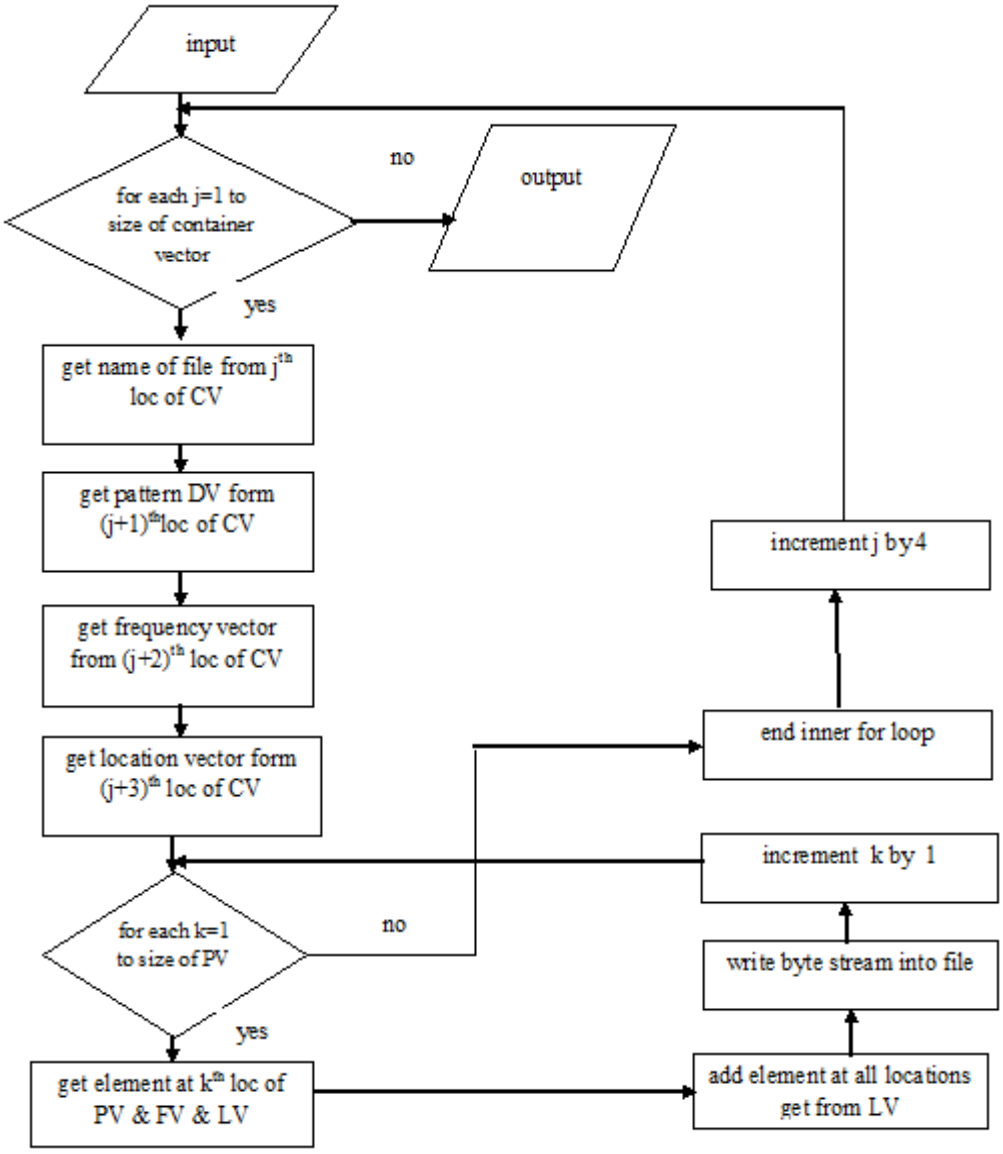


Figure 4.10 : Flow Chart of PDPA

Output: - The original regenerated files that contain the pattern given by client.

4.5.4 Pattern Based Processor Agent (PBPA): - processes the Result Container prepared by the PBSA during the itinerary. It retrieves three vectors from the Result Container (*document vectors*¸ *pattern vector* and *frequency vector*) for every visited node in the itinerary. By processing these vectors it regenerates all the original files and stores at local client node.

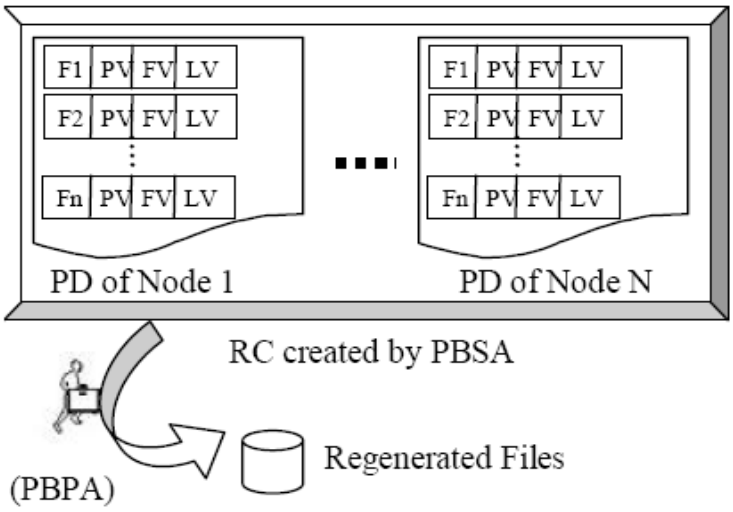


Figure 4.11: Working of PBPA [14]

Algorithm- Pattern Based Processor Agent

Input: A container vector that contains the name of each file and its corresponding byte stream

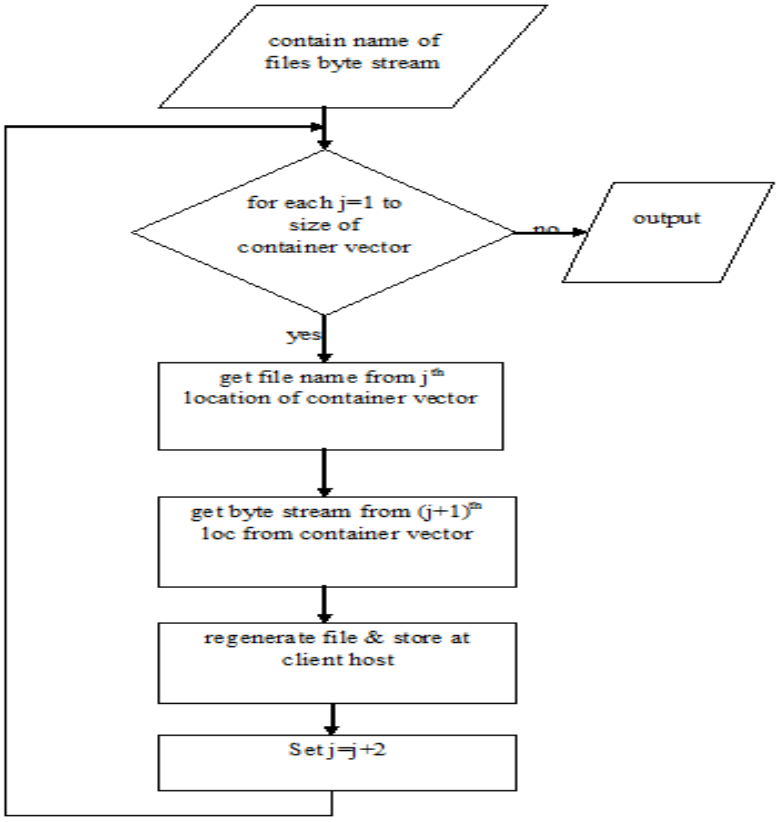


Figure 4.12 : Flow Chart of PBPA

Output: The original regenerated files that contain the pattern given by client.

CHAPTER 5

IMPLEMENTATION AND RESULT

The PMADE platform has been implemented on an Intel Pentium D2C T2390 1.86 GHz, RAM 3.00 GB, and 32 bits operating system, running on Windows Vista™ Home Basic. The MA paradigm used is based upon.

5.1 Implementation: - Here we examine the experiment on the single host. There are three host:-

- Server
- SecurityServer and
- Client.

The very first step is to start the server and then the SecurityServer and then the client. The following screen snap-shot shows as the starting of the server.

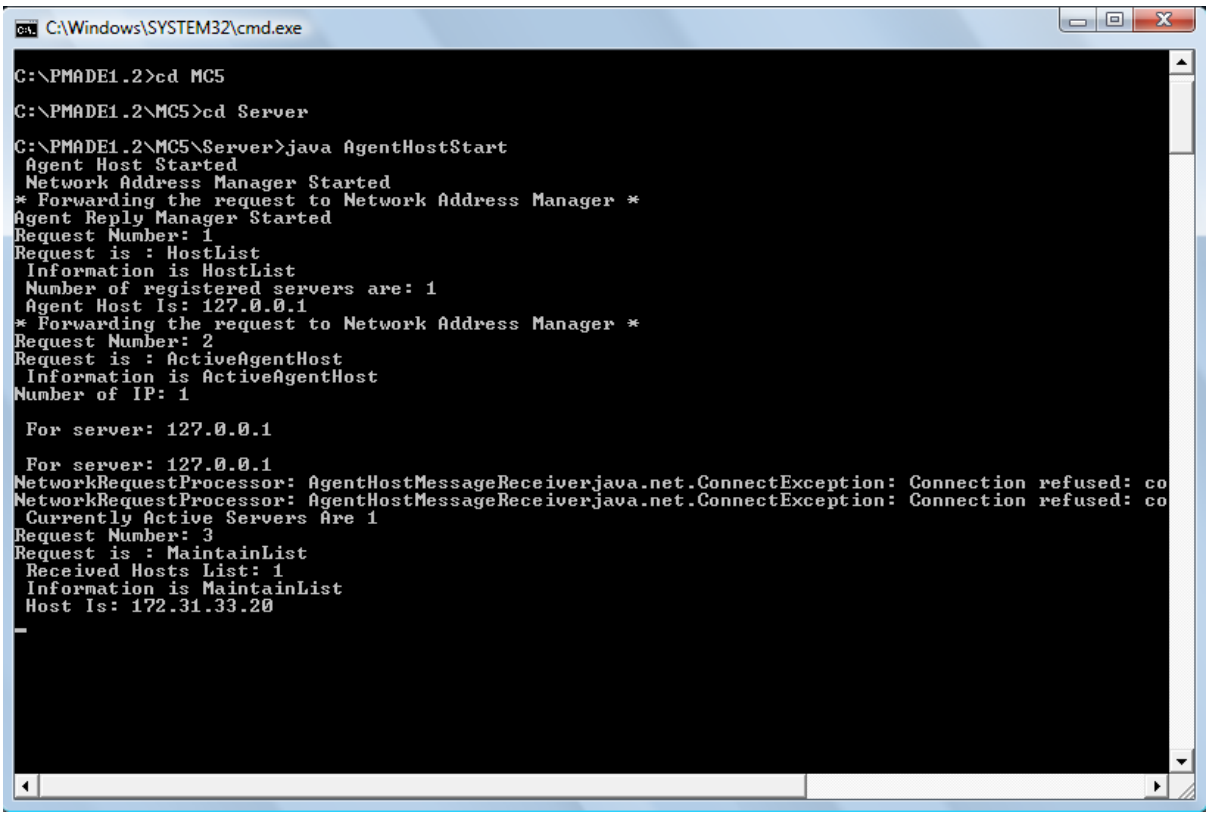


Figure 5.1: Server

Now, the SecurityServer verifies it's the authentication of the user and provides a token to user. The following screen snap-shot depicts when the server has been authenticated.

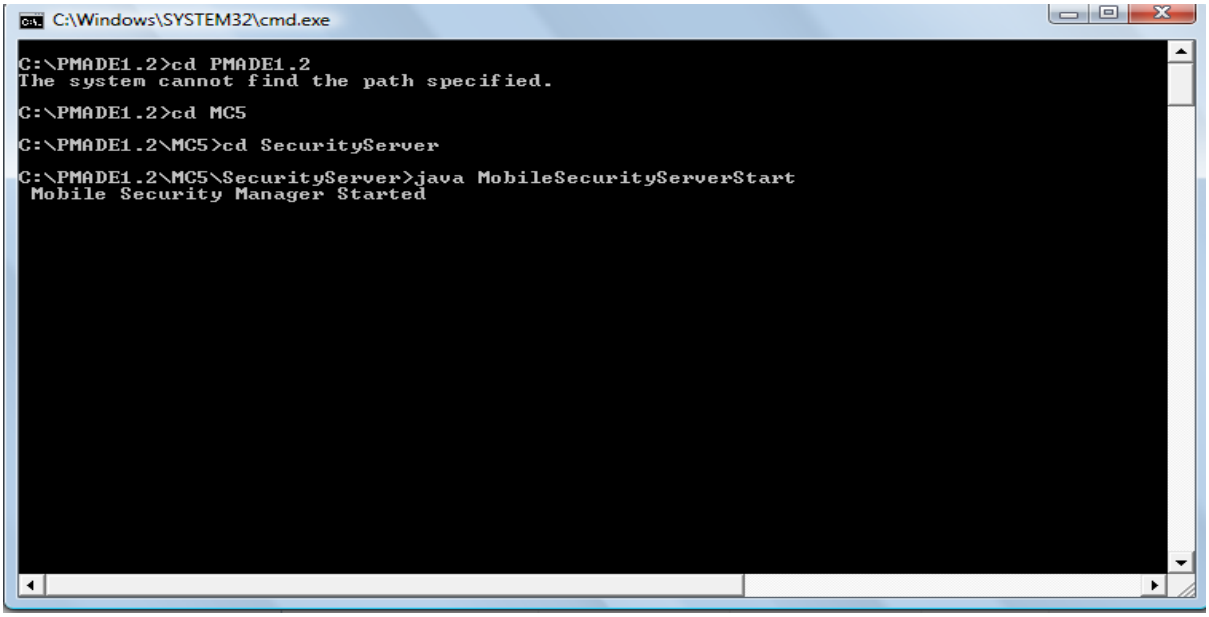


Figure 5.2: SecurityServer

Lastly, the client starts. The following screen snap-shot shows the message "Result Manager started" it mean our system has been set up properly and the client or user are ready to register to communicate with system.

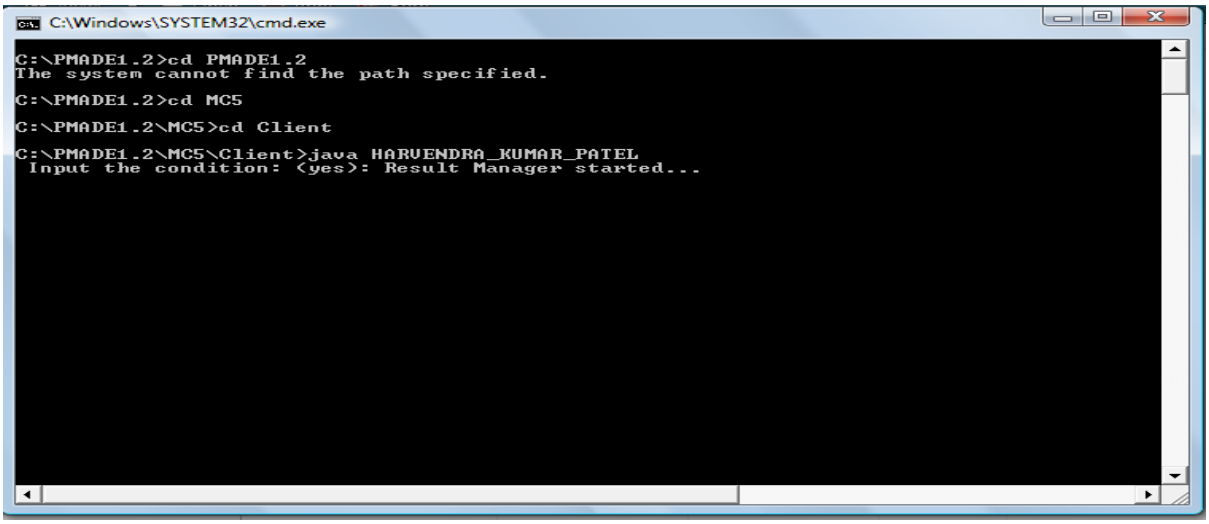


Figure 5.3: Client

To take care of the security aspect of the user, the client/user has to register itself with SecurityServer. There are three different parameters, on behalf of these parameters DM can be done, those parameters are :-

- User
- Mobile
- Receive

Figure 5.4: Client Activity during data mining

User: - In this option user or client gets an acknowledgment which is issued by the security manager. This is done for security purpose so that unauthorized client cannot send the

malicious code. As the user has been authenticated then it is ready to send (or launch) the agent on the networks.

Mobile: - In this option, the agent code name, version of the code, itinerary of the code, is given as inputs and the number of the host in the itinerary that have been visited, round trip time and finally the output is depicted as output.

Receive: - In this option, we are ready to gather the result from different nodes based upon the agent code name and version of the agent. The results are stored in the pdga-pdpa folder.

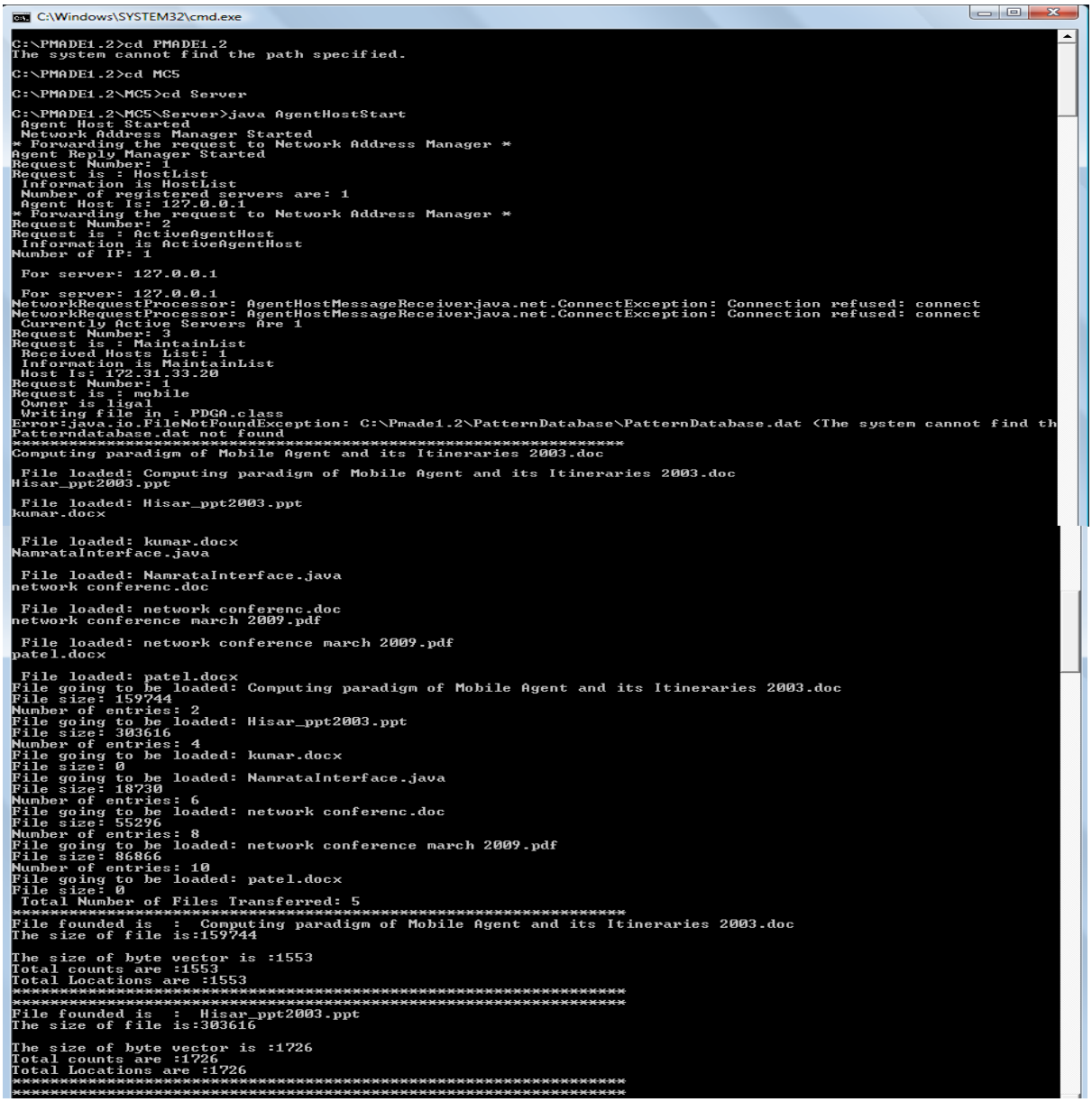


Figure 5.5: Server Activity during data mining

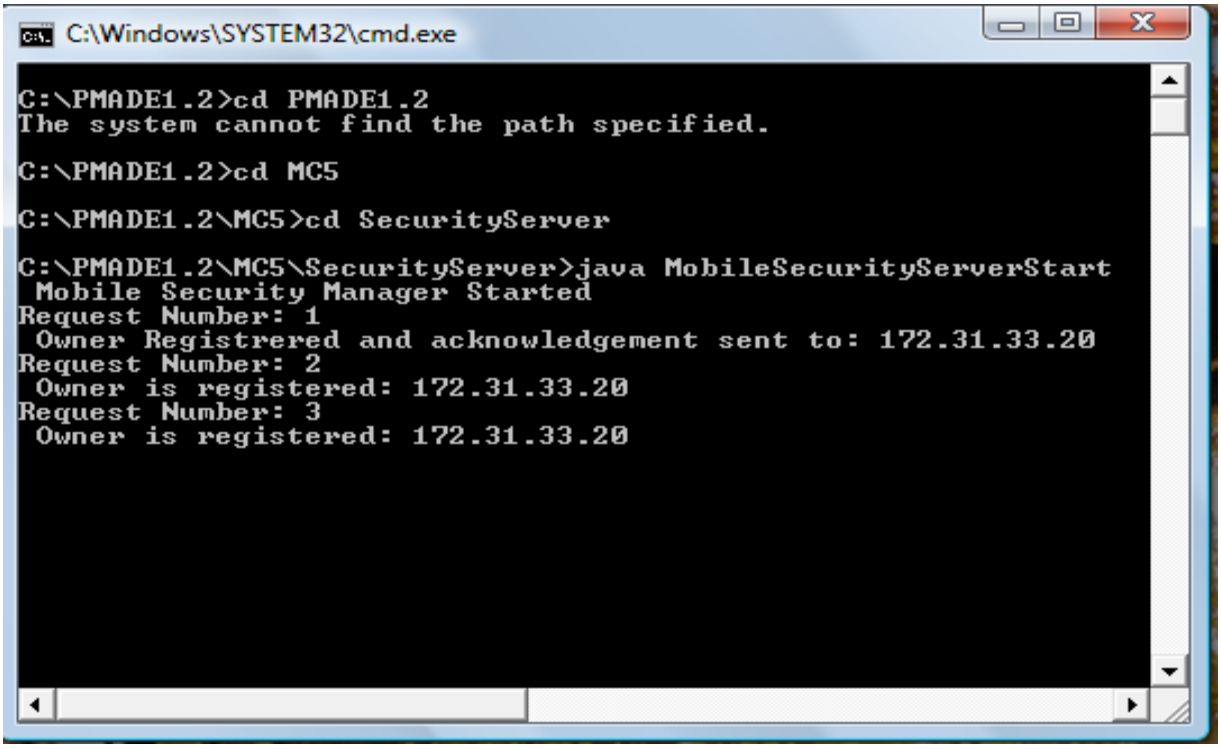


Figure 5.6: SecurityServer Activity during data mining

The agent trip time depends on the size of the gathered data or mined data. As increase the data agent trip time is also increase. Network traffic also play the major role, due to this reason, we have performed this analysis when network is very lightly loaded and busy and taken average case. The performance of the PDGA changes as collected data size changes. When large size of data is collected the Agent Trip time increases. When we compare the performance of both the agents (PDGA and PBSA), then the Trip time in case of PDGA is larger than that of in the PBSA because the computational time (computations in processing the files for generating the Pattern, frequency and locations document vectors) of PDGA is more as PBSA. So the agent trip time in case of PDGA comes out to be larger. But the advantage of using the PDGA over the PBSA is that the PDGA consumes less network bandwidth. The transfer's documents will be compacted in the size; rather than the complete data of the file.

CHAPTER 6
CONLUSION & FUTURE SCOPE

Conclusion and Future Scope: -

In this thesis, we presented a multi-agent system for data gathering using mobile agent. We have identified four agents that are PDGA, PDPA, PBSA and PBPA for four different operations. Two are mobile agent (PDGA and PBSA) and two are stationary agent (PDPA and PBPA). Other system components (Policy, Agent-Agent, Interface, and Agent-Agent Communication layer) are provided in the framework to help the MAs to improve their performance as well as their autonomy and proactively for the navigation through the agent system infrastructure. For the evaluation, our focus was on pattern matching.

Researchers all over the world are applying this novel paradigm for solving various problems in a distributed environment. We use the best algorithm to mobile agent (PMADE) so that it retrieves the data in fastest manner. We find better itinerary to the mobile agent so that it could take minimum time to retrieve the data.

In this thesis we have carried out a detailed analysis of PMADE architecture and mobile agent environment. The biggest of such computational paradigms is that code moves to the data instead the data moving to the code, although this technology is in its infancy stage and improvements can be made to make it more efficient in terms of time requirement and usage. The main limitation of mobile agents is the security risk involved in using mobile agents. Firstly, a malicious mobile agent can damage a host. (For example, a virus can be disguised as a mobile agent and distributed in the network causing damage to the host machines that execute the agent) and on the other hand a malicious host can tamper with the functioning of the mobile agent. The following are some more limitations of the mobile agent paradigm:

- More complex the setup then client/server.
- Everything can also be done with client/server
- It is platform dependent, where it runs platform should be needed.

The proposed model in current work provides a mobile agent based software retrieval service that provides software developers with a mechanism to search and retrieve the required reusable component from various repositories at different locations i.e.

distributed repositories. Although this service can be used for fixed computers it can be extended in a mobile environment although emphasis on optimizing the use of battery and wireless communications, would be required.

REFERENCES

[1] C. Ghezzi and G.Vigna “Mobile Code Paradigms and Technologies: A Case Study in Proceedings of the First Int. Workshop on Mobile Agents"(MA97)” Berlin, Germany, April 2003.

[2] Tripathi, R., T. Ahmed and N.M. Karnik “ Experiences and future challenges in mobile agents programming” Microprocessors and Microsystems, 2001.

[3] Aridor, Y., & Lange, D.B. (1998). “Agent design patterns: Elements of agent application design” Proceedings of the 2nd International Conference on Autonomous Agents. Minneapolis/St. Paul, USA. 108 -115.

[4] Nwana, Hyacinth. “Software Agents: An Overview.” Knowledge Engineering Review, Vol.II.

[5] D. Johansen, “Mobile Agent Applicability,” Proc. MA ’98: 2nd Int’l. Wksp. Mobile Agents, Lect. Notes in Comp. Sci., no. 1477, Springer- Verlag, 1998.

[6] iisc.ernet.in/ academy/ resonance/ July2002/ pdf/ July2002 p35-43. pdf.

[7] softwareresearch. net/ site /teaching/ WS0203/ PDFdocs.DS/ mobile_agents.pdf.

[8] Byung-Hoon Park and Hillol Kargupta, “Distributed Data Mining: Algorithms, Systems, and Applications”, University of Maryland Baltimore.

[9] R. B. PATEL, K. GARG,”A Comparative Study of Mobile Agent and Client-Server Technologies in a Real Application”, ADVANCES IN DATA MANAGE-MENT 2005 Jayant Haritsa, T.M. Vijayaraman (Editors) © CSI 2005.

[10] Lange D.B. and Oshima M. “Seven good reasons for mobile agents” Comm.ACM, 42(3):88–89, March 1999.

[11] Robert S. Gray. Agent Tcl:“A transportable agent system” ,In James Mayfield and Tim Finin,editors, Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Manage-ment (CIKM 95), Baltimore, Maryland, December.

[13] Carlos Lucena José Alberto Sardinha Alessandro Garcia, Jaelson Castro Centro de Informática, Alexander Romanovsky, Paulo Alencar Donald Cowan, “Software Engineering for Large-Scale Multi-Agent Systems – SELMAS’2003”.

[14] Gurpreet S. Bhamra, R. B.Patel, A.K.Verma,"An Agent Enriched Distributed Data Data Mining On Heterogeneous Networks" in IEEE ICACS 6-7 march 2009.

[15] Karlsson, Bjorn, Oscar Backstrom, Wlodek Kulesza, and Leif Axelsson. "Intelligent Sensor Networks - an Agent - Oriented Approach." REALWSN, 2005. <http: // www.crl.se/ publications/ IntelligentSensorNetworks - anAgent-Oriented Approach_paper.pdf>.

[16] Picco, G.P."Mobile Agents: An Introduction" Microprocessors and Microsystems 25, 2001, 65-74.

[17] Tripathi, R.. T. Ahmed and N.M. Karnik, " Experiences and future challenges in mobile agents programming", Microprocessors and Microsystems, 25, 2001.

[18] Patel, R.B. and K. Garg, "A new paradigm for mobile agent comp",uting, WSEAS Transaction on Computers, 1, 2004, 57-64.

[19] Sun Microsystems, Java Object Serialization Specification, 1997.

[20] Dimitri Verspecht "Mobile agents for mobile platforms"Vrije Universiteit Brussel, faculty of Science, Department of computer science, 2001 -2002.

[21] IBM Aglets website: http://www.trl.ibm.com/aglets - creators of Aglets.

[22] G. S. Bhamra, A.K. Verma, R.B. patel, " Extracting pattern from the distributed data sources in open network: An agent approach", International J. Of systemic, Cybernetics and Informatics, July 2008.

[23] R. B. Patel, Namrata, " A MOBILE AGENT FRAMEWORK FOR CONTEXT-BASED SEARCHING OVER THE INTERNET ", International Conference On Advances In Information And Communication Technologies, Manipal Institute of Technology, Manipal , Karnataka 28-30 December, 2007, pp. 133-140.

[24] Harvendra Kumar, A.K. Verma, Ajay Kumar "Comparative Study of Distributed Computing Paradigms", National Conference in Amritsar Punjab Impact of IT on Society-Emerging Trends and Issues Feb 28- March 01,2009.

[25] Harvendra Kumar, A.K. Verma, Ajay Kumar " Computing paradigm of Mobile Agent and its Itineraries" National Conference", 24 March 2009.

[26] YANG Gong-ping, ZENG Guang-zhou, " Mobile Agent Life State Management" IMACS Multiconference on "Computational Engineering in Systems Applications"(CESA), October 4-6, 2006, Beijing, China.

[27] Hairong Qi, Feiyi Wang,"Optimal Itinerary Analysis for Mobile Agents in Ad-Hoc Wireless Sensor Networks".

[28] Rajwinder Singh , Navdeep kaur , and A.K.Sarje " Designing Secure Itineraries Protocols for Mobile Agents" , IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.12, December 2006.

[29] VuAnhPham & Ahmed Karmouch, " Mobile Software Agents: An Overview", IEEE Communication, July 1998, pp. 26-37.

[30] Deitel & Deitel, "Java: How To Program", Prentice Hall, 2000.

[31] R. B. Patel, V. K. Katiayar, Vishal Garg, "Mobile Agents in Wireless LAN and Cellular Data Networks, Journal of Computer Science, Science Publications New York, USA, 2(5): 410-418, 2006.

[32]. Deitel & Deitel, "Java: How To Program", Prentice Hall, 2000.

[33]. R. B. Patel, V. K. Katiayar, Vishal Garg, "Mobile Agents in Wireless LAN and Cellular Data Networks, Journal of Computer Science, Science Publications New York, USA, 2(5): 410-418, 2006.

[34] Chess, D. M., Harrison, C. G., and Kershenbaum, A. Mobile agents: Are they a good idea? Research Report, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY. Mar., 1995; www.research. ibm.com/iagents/publications.html.

[35] VuAnhPham & Ahmed Karmouch, "Mobile Software Agents: An Overview", IEEE Communication, July 1998, pp. 26-37.

[36] Patel, R.B. and K. Garg, A new paradigm for mobile agent computing, WSEAS Transaction on Computers, 1, 2004, 57-64.

[37] R. B. Patel and Neeraj Goel, " Mobile Agents in Heterogeneous Networks: A Look on Performance", J ournal of Computer Science 2 (11): 824-834, 2006.

[38] Programming and deploying java mobile agents with aglets.

[39] Platform for Mobile Agent Distribution and Execution.

LIST OF PUBLICATIONS

1. Harvendra Kumar, A.K. Verma, Ajay Kumar ”Comparative Study of Distributed Computing Paradigms”, National Conference in Amritsar Punjab Impact of IT on Society-Emerging Trends and Issues Feb 28- March 01,2009.
2 Harvendra Kumar, A.K. Verma, Ajay Kumar “ Computing paradigm of Mobile Agent and its Itineraries” National Conference”, 24 March 2009.
3. Harvendra Kumar, A.K. Verma, “Analysis of Distributed Computing Paradigms” International j. Of information Tech.