Study of Mobile Agent Server Architectures for Homogeneous and Heterogeneous Distributed Systems

Rahul Singh Chowhan
M.E. Scholar, Dept. of C.S.E.
M.B.M. Engineering College
Jodhpur, India

Rajesh Purohit, PhD
Professor and Head, Dept. of C.S.E.
M.B.M. Engineering College
Jodhpur, India

ABSTRACT
Mobile agents are becoming pre-eminent by not only outperforming in comparison with the conventional techniques such as RMI, RPC etc. but also by surpassing their loopholes. They promise to solve many major issues of high network bandwidth consumption during communication, bottleneck problem of centralized system, even can act as intrusion detection agents, and may also be used as monitoring of various nodes in multifarious domains like e-commerce services, for load balancing in cluster, health care monitoring systems, air traffic control systems, and many more. In this paper, the agent server required to allow mobile agents on any machine in network are compared for homogenous and heterogeneous nodes. The homogeneity and heterogeneity of nodes is defined at the hardware level and type of OS installation. Basically, a mobile agent is moving the code to data rather data to code. Agent and agent server are two different parts, in which agent is a computational, operational and communicative entity while the agent server takes care of fundamentals execution and security features. To all intents and purposes, these agent servers help mobile agents to interact and engage with the underlying system acting as an execution environment for them. Agent servers, also called as agency or agent runtime environment, may differ for different platforms and this contrast lies in the software architectural components which they contribute being a middle layer in between the mobile agents and system platform. This paper focuses on architectural dissimilitude between agencies of heterogeneous and homogeneous distributed systems.

Keywords
component; mobile agents; software architecture; distributed systems; agent migration; marshling; agent trasnfer protocol

1. INTRODUCTION
Mobile Agents are traveling intelligence that adds up a new level of abstraction between clients and requested resources, yet maintaining the autonomy and transparency while communicating at remote hosts. This notion of mobile agent has influenced diverse field of disciplines like: computer science & networks, information technology, machine learning, object-oriented programming, applied artificial intelligence, robotics & software engineering, human-computer interaction, distributed and parallel systems, mobile & control systems, data extraction and mining, decision-making systems, information retrieval, e-commerce, big-data management, SaaaS and many more [1][2].

Any mobile agent platform has segregation into: mobile agent and runtime environment. The mobile agent is the software code on move while runtime system is the mobile agent server that approves the execution, communication, migration and cooperation of mobile agents. Each runtime environment runs atop the operating system as a middleware. It serves the execution environment for agent programs running itself on the top of the Java virtual machine (JVM) specified for the particular machine.

The software architecture of mobile agent server must continue to abide on all network nodes to which agents can accost. This means that mobile agent can be allowed on any node for execution only after installation of mobile agent server [3]. This serves the mobile agents with a runtime environment on the current machine. When a mobile agent needs to migrate from current context to destination context, it simply requests the local runtime system then runtime system migrates the mobile agent to new runtime environment carrying its data state, execution state and implementation along with it [4].

Broadly, the requisites to concede the mobile agents on remote hosts or across the network of connected nodes for communication, migration, storage and security purposes are as follows [5]:

- Standard Execution Platform/Language
- Agent Persistence
- Inter-agent and Agent-agency Communication Mechanism
- Cooperativeness and Collaboration in Multi-agent Systems
- Agent Authorization and Security

Apart from above mentioned the mobile agents also need the mobile agent server/agency for local interaction with the underlying execution environment [6]. In distributed systems, clients can have different capabilities of resources available with them it might also happen that clients are not stationary at a location. Secondly, the client hosts might be situated or has moved to a distant location with less network connectivity. [1]

Customarily, mobile agent is defined as a self-sustaining and standalone software entity which does its work on behalf of a user. An agent may even continue to run involuntarily regardless of its owner's direct connectivity and is capable to execute on multiple nodes in the network. Substantially, this mobility feature further improves the outcome of each computing element participating in the network. This enables the whole computing domain to handle a number of tasks. Mobile agents have been put forward as a middleware technology to deal with challenges like high latency, intermittent connections, asynchronous and autonomous execution, huge transfers and disconnected operation. [7]

2. AGENT EXECUTION PLATFORM AND TRANSFER PROTOCOL
Agent execution platform comprises of agent server that is also runtime environment and container for mobile agents. It allows the management, monitoring, execution and other life cycle events of all mobile agents residing on the system [8]. In single agent system, many agents can execute independent of each

other whereas multi-agent runtime system allows execution as well as communication of multiple agents with each other. The existing agent execution platform also provides with some kind of communication mechanism like request/reply mechanism, publish/subscribe mechanism [9], push/pull mechanism [10], etc. Each agent can access the fundamental methods by invoking APIs provided by runtime system it is residing on currently. The agent uses the go command only rest of required functioning is carried by the agent server. It halts the execution of agent, serialized and marshals it to send it to destination via network communication protocol like TCP/IP, HTTP etc. On the receiver side, agent reconstituted and started again. Apart from this execution platform may also have introduce a mobile agent tracking and locating techniques such as central servers scheme, broadcast/multicast scheme, forwarding pointer scheme and hierarchical scheme [11].

The transport layer takes care of migration of an agent from current host to the destination host in stream of bytes that contains class implementations as well as the serialized data/execution state of the mobile agent. This layer is specified as an API (Application Programmable Interface) and is called as Agent Transfer and Communication Interface (ATCI) [9] [12].

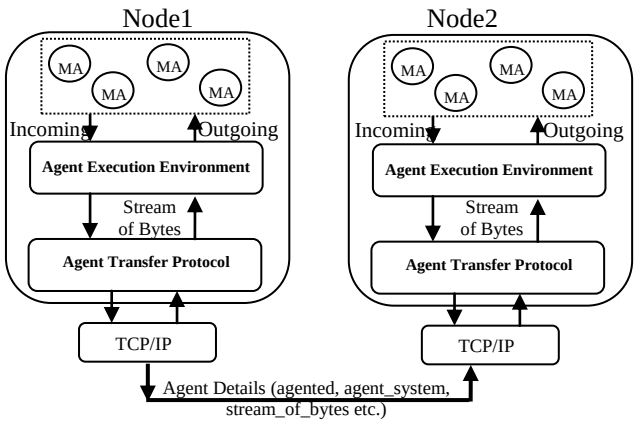

Figure-1 Agent Migration Parameters

It qualifies the agents' runtime environment/agent server to use the transport layer independent of protocol. The implementation of ATCI is responsible for establishing a communication between agents as well as their sending/receiving on current context. The current Agents implementation uses the Agent Transfer Protocol (ATP) [13].

ATP is an application-level protocol for transmission of mobile agents in distributed agent-based systems. ATP is modeled on the top of HTTP protocol which is used to deploy the content of an agent in an agent-system-independent manner. ATP proffers a simple, platform-independent and individualistic protocol for transferring agents between networked computers and also supports message passing. Mobile agents may be programmed in many different languages and for diverse vendor specific agent runtime environment [14].

ATP extends the scope of employing agent mobility in a general and uniform way:

- A machine hosting agents has an ATP-based agent service, which is a component capable of receiving and sending agents from remote hosts via the ATP protocol. The agent service is identified by a unique address, independent of the specific agent platforms supported by the machine. A machine can run multiple agent services.
- The agent runtime system supports the migration of the agent, halting its execution and then marshaling the agent data items to the destination via the underlying communication protocol, e.g., TCP channel, HTTP (hyper text transfer protocol), and SMTP (simple mail transfer protocol). [13]
- A machine can host different types of mobile agents, provided it supports the compatible agent platforms. Any agent platform should include a handler of ATP messages.
- An ATP message carries sufficient information to identify the specific agent platform at the receiver host so that ATP handler can be called to handle the message.
- The ATP protocol works in a request and response manner in between ATP services.
- ATP service A establishes a connection with ATP service B then sends a request to B and waits for the response. Thus, A and B act as a sender and receiver respectively. An agent host machine can support more than two agent systems, which may be provided by different vendors.
- A request includes a request line, specifying the request method, the protocol version, and the required resource, followed by a MIME-like message containing request modifiers, sender information, and possible content in its body.
- A response includes a status line that specifies the protocol version, a success or error code ensued by a MIME-like message containing response modifiers, sender information, and possible content in its body.
- The implementation of ATP needs no creation of a new thread altogether to handle an incoming connection request. Instead, ATP does thread pooling and it simply puts out an available thread from it to assign it to client request. This possibly allows handling multiple requests efficiently and every agent has its own threads of control or execution thread [15].

3. AGENT MARSHALING AND MIGRATION

Mobile objects and their data values from a running entity cannot be straightaway transferred over the network to other machines. Firstly, the transformation of internal data on the current machine is required to be represented in external data form such as in the string of bytes or in the binary form. That is why, marshaling and un-marshaling of mobile agents is required to be carried out before their migration [16].

Marshaling is the process of collecting memory representation of an object data and assembling them into a data format suitable for transmission or storage purpose. Typically, this is carried out when object or data is required to move from one program to another, within different parts of program or in between the machines. While process of un-marshaling happens on the other node that acts as a recipient. This involves the disassembling of received data to produce an equivalent collection of meaningful data items at the recipient. The conversions are possibly achieved through serialization and de-serialization of object on current and remote hosts. Serialization and marshaling has a thin line lying as difference between them such that marshaling can record the state and codebase, i.e. an URL list needed to load the object code, of local as well as remote object while serialization happens to convert the marshaled object into stream of bytes but codebases and remote object cannot be serialized [17]. This process is shown in figure-1.

The process of marshaling and un-marshaling the agent object is carried out by mobile agent server/agent runtime environment. The sender mobile agent server marshals the mobile agents and transfers it to receiver mobile agent server using via media or communication channel, then the mobile agent server at the at receiver end, receives the bytes and un-marshals the mobile agent. This sequential process of transferring the mobile agent is called as mobile agent migration.

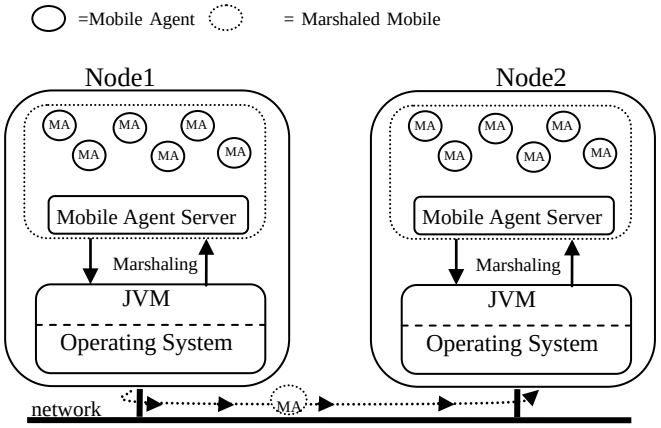

Figure-2 Migration of Agent between Node1 & Node2

4. PROPERTIES OF MOBILE AGENT AND AGENT SERVER

4.1 Notions of Mobile Agent Migration on Agent Server

Both the terms "mobile" and "agent" are of Latin origin which means "something that's on move" and "react to situation" respectively. On the other side, code moves sluggishly when transported on a diskette or tape. In its most primitive form, somewhat faster, code was moved via the FTP mechanism but it still involved manual intervention of user.

Examples of this type of migration are remote batch jobs and the use of postscript to control printers. Fairly fast channel is www; again it requires active participation of user. The next shift happened with the arrival of Java applets and JavaScript in www. They keep the network connection alive even when the user is inactive. Email is quite a fast channel because the emails are handled automatically by the mail servers, and one email can travel around the world in a couple of minutes. For sure, the fastest channel is direct TCP/IP, or UDP, which can be even faster in some cases for example in live video streaming where security is of not much concern [17] [18].

Agent migration in code mobility involves two notions to allow the migration of mobile agents to and from mobile agent server: Weak Migration and Strong Migration [19].

1. Weak Migration

a) When moving, a mobile agent carries code implementation and data state.

b) Data State: global or instance variable.

c) On moving, execution has to start from the beginning. Runtime environment needs to call the methods explicitly to restart the execution of agent all over again.

2. Strong Migration

d) When moving, a mobile agent carries code implementation, data state and its execution state.

e) Data State: global or instance variable.

f) Execution State: program counter, local variables, saved registers and threads.

g) Before moving to new host, agent is suspended, marshaled, trasnferred, unmarshaled and resume its execution continuing from the point it stopped on the previous host.

4.2 Executability Feature

Any business logic or data written in specific programming language that is understandable by underlying system via an IDE can be called executable for that environment. There are degrees of executability involved that includes:

1. **Text** contains information that is readable by humans only. An AI program might understand it, but it contains no structured information.
2. **Marks** can be instructions inside the text denoting how to interpret it. Usually, this is just a few characters, such as how to write a word in bold text. In LaTeX it is \textbf{\textit {Marks}}.
3. **Macros** are instructions that can generate a piece of text. For example, a macro like <date> would generate today's date and insert it into the text.
4. **Controls** are instructions for how data should be interpreted. They can declare conditional text, define variables, and change them. Basically, they have taken over the control of the interpretation of the data.
5. **Scripts** are written in complete interpreting programming languages, which are Turing machine equivalent. To put it simpler, any program can be made in this language. The execution point can be controlled with loops and jumps. They can even access many system functions.
6. **Byte code** is virtual machine code that is interpreted in a virtual processor. At this point the code is really not readable by humans.
7. **Machine code** is instructions for the processor in the computer. Machine code is often the result of a compilation. This level is usually the lowest, even though in some processors it is possible to write micro codes, which are instructions for the gates and subsystems inside the processor.

4.3 Dependency of Agent Autonomy on Mobile Agent Server

Once the mobile agent is allowed to run on machine via mobile agent server it can start acting on its own without any outside control. It is responding, reacting, or developing independently of the environment autonomously. A mobile agent is self-contained while navigation but for it is dependent on the underlying server for communication, computation and execution. A mobile agent should be aware of its itinerary, i.e. about all participating nodes, when transporting through the heterogeneous and dynamic arrangement of network nodes and where to go to look for available resources in the network. A piece of programming code and data is enclosed as an encapsulated entity that contains implementation, data and execution state in single unit. The execution state is the current state of mobile agent with its variables and bindings to resources for example it may be a reference to a database or a local printer [20].

Degrees of agent autonomy include:

1. *Mobile Autonomy:* It is the capability of self-navigation using navigation modes: serial or parallel,

decision making and docking at time of unreachable destinations or broken connections, like disconnection of a laptop etc., of mobile agents through the underlying runtime environment. Whenever there is an availability of multiple resources, a mobile agent can clone itself and assign a subtask to each of its duplicates for execution of a distributed task concurrently. The division of a task into multiple subtasks in form of agents can benefit the whole system with simultaneous and co-existential execution of subtasks. Finally, the results from various subtasks of different clones of mobile agents is combined as one result. So autonomous mobile agents are network-aware entities that are dynamic and reusable on which Web applications can be constructed.

2. *Computational Autonomy:* It is that a mobile agent can get enough computational functions making use of all kinds of computational resources to accomplish an assigned distributed task. A mobile agent can call for various functions residing on its visiting nodes via mobile agent server into its process to execute them when required. Even other way round, a mobile agent can as well complete a distinctive computational task by executing its own carried functions on current agent server when visiting a network node with desirable resources.

Communicational Autonomy: This allows a mobile agent to send and receive messages in an asynchronous, anonymous and indirect way. This is more the property of mobile agent server than to the mobile agent itself. Mostly, communicational autonomy is high in multi-agent systems as they require inter-agent and agent-agency communication to accomplish cooperative tasks in distributed environment. Secondly, Agent-hosting execution environment interactions are required when a mobile agent make use of resources and services in a local environment of network nodes it visits [21][22]. ATP defines following four standard request methods for agent services:

1. *Dispatch an Agent:* To dispatch an agent from an agent service A to an agent service B, A sends a dispatch request to B. The agent goes along with the request. When B receives the agent, a response to A will be send containing a status code.
2. *Retract an Agent:* To retract an agent from an agent service B to agent service A, the latter calls a retract request method to B. The acknowledgement from B is actually a response with a status code and the specified agent in its body.
3. *Fetch a Class:* For execution of an agent, the agent's origin i.e. the service A needs to send an executable code to agent service B. To achieve this, B initiates a fetch request to A. The reply from A is a response with a status code and the required executable code in its body.

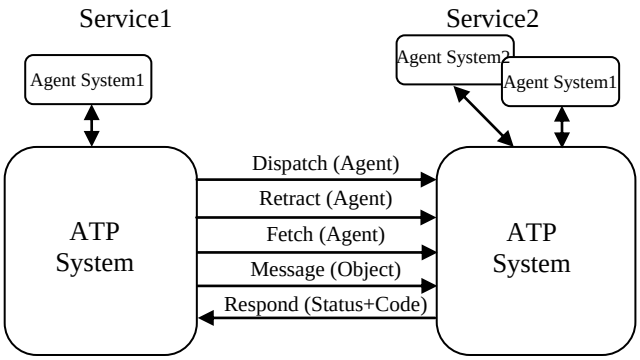

Figure-3 Standard Methods for Agent Services

4. *Send a Message:* To send a message from an agent in agent service A to an agent in agent service B, A sends B a message request, containing that message. The reply from B is a response with a status code.

5. ARCHITECTURAL VIEW OF MOBILE AGENT SERVERS

Mobile agents need a platform to run over any host in a heterogeneous or homogeneous network. This is a software program at each site which handles the incoming and outgoing agents dealing with their execution and is often called an "Agency". The agency is a mobile agent server which can be built differently depending on which type of agent system is needed and every node in the network must have this server. Agency is also responsible for sending messages between agents and does some authentication if necessary. The heterogeneity and homogeneity of mobile agent server is dependent on hardware configuration and operating system of the underlying machine. In homogeneous system arrangement all system has similar resource capabilities so no prior information is required about the hardware resource availability is required to figure out. But the current state or load information of machine is necessary to be known to use it for various purposes like load balancing, e-commerce etc. This information is collected by using monitoring mobile agent. In heterogeneous system arrangement, the node capabilities are different for every node and not known as prior information in homogeneous system arrangement. This scenario may also include thin and thick clients with varying configuration.

The generalized software architecture of mobile agent server for homogeneous distributed systems has various components that include:

1. *Communication Module:* It handles incoming and outgoing agents, as well as the messaging between non-local agents. There can be security functionality too, such as encryption/decryption mechanism to encrypt outgoing agents and decrypt incoming agents.
2. *Repository Module:* It performs authentication, also sets priorities and queues up agents for later execution.
3. *Executer/ Interpreter Module:* It has an interpreter and can sometimes run agents written in different languages.

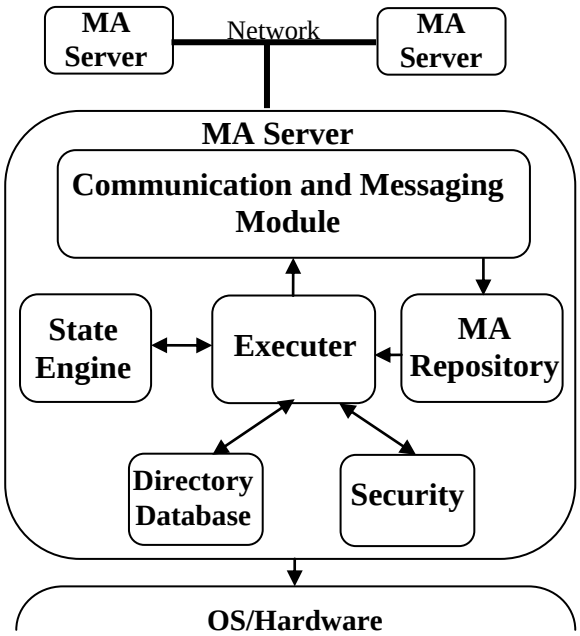

Figure-4 Mobile Agent Server for Homogeneous Systems

4. *State Engine:* This contains the current state of the agency and can have some kind of rule or inference engine that decides what to do with the agents. It also handles local inter-agent communication.
5. *Directory Database:* Data about agents are stored or retrieved in this directory. Also agents can access the database to write their state status.
6. *Security Module:* This act as a kind of sandboxing environment that keeps track of what the agents are allowed to do. It also monitors the agency for any malicious activities.

The mobile agent server for heterogeneous systems allow various types of client that may be thin or thick client. The client capabilities are not known prior to the submission of mobile agent, so for client side information transport channels such as email or HTTP-download. The architectural components of mobile agent system for heteregeneous systems include [22]:

1. *Agent Initiator:* For the first time when client connects to the agent server it needs to share the client device capabilities in terms of execution memory, CPU speed, storage and installed version of JDK, JRE and JVM to allows serialization of Java based mobile agents. The installed initiator follow installation, configuration & initiation of initiator, and installation of agent client. If any characteristics of the client are changed like installing a new version of JVM or the agent system itself changes then agent server may initiate a client update process. So with the new installations mobile agent system could become more CPU and/or memory efficient than the obsolete version.
2. *Services:* Tracking of all services like registering new agents, maintaining life cycle events, taking care of execution etc are maintained at this level. This also takes care of serialization and deserialization of agent objects by activating and deactivating them from underlying persistent storage. With object serialization, mobile agents can save and load the state of objects to the disk or over a network which can be restored at a later time, and even a later location. Object being available on persistent storage, can be moved from one computer to another in a network maintaining its state.
3. *Agent Joiner/Splitter:* This module works only for the thin clients. These are the clients that don't have enough capabiliies to run the mobile agent server locally. Joiner/splitter does this by seperating the code part from data out of which data part moves on with mobile agent while the code part is remains on the server machine's agency which is reconstituted on arrival on mobile agent.
4. *Agent Dispatcher:* The job of dispatcher is to send off the mobile agentss to the respective destinations. As connections proliferate, network topologies necessarily become more and more dynamic. Devices may move from place to place, or maintain intermittent connections, or change their relationships to the network and their peers on the fly. Mobile Agent dispatcher may use dynamic routing algorithm which is applicable in current networks with a large scale, a large amount of switching nodes and high traffic. Dynamic routing uses a dynamic routing protocol to automatically select the best route to put into the routing table.

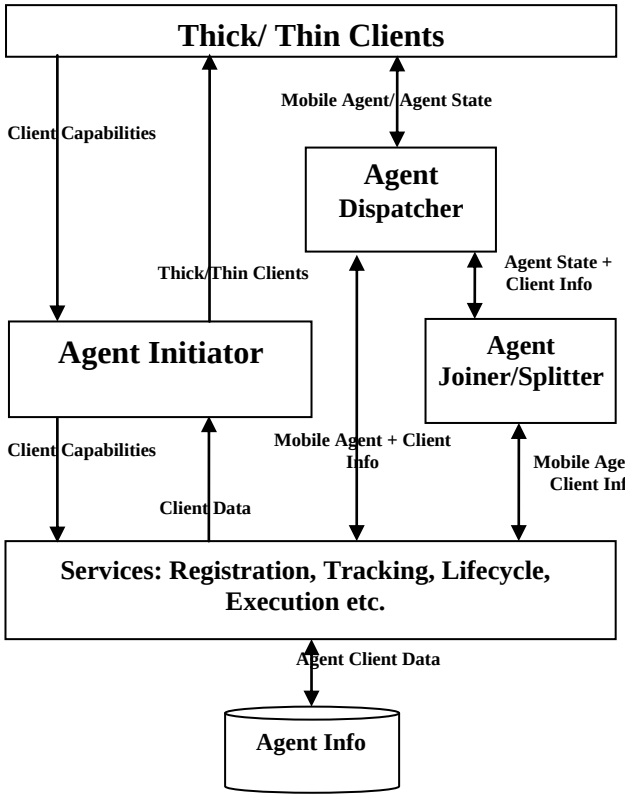

Figure-5 Mobile Agent Server for Heterogeneous Systems

6. CONCLUSION

Mobile agents are functional in various fields of distributed and parallel environment based applications like information retrieval, advanced web search, data filtering, e-commerce, etc. The real-time convention of free roaming mobile agents involves numerous mechanisms to inform server and that too reducing the server communication overhead. Mobile agents have been starting to be widely accepted in various domains because of their autonomous execution and mobility feature. For many properties like security, execution, communication etc, they are dependent on the mobile agent server. This server

for homogeneous and heterogeneous distributed systems may vary as per the client capabilities available with the machines.

7. REFERENCES

[1]. Youssef M. Essa, Gamal Attiya, and Ayman El-Sayed. "Mobile agent based new framework for improving big data analysis." In Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on, pp. 381-386. DOI**:** 10.1109/CLOUDCOM-ASIA.2013.75, IEEE, 2013.

[2]. Gaoyun Chen, Jun Lu, Jian Huang, and Zexu Wu. "Saaas-the mobile agent based service for cloud computing in internet environment." In 2010 Sixth International Conference on Natural Computation, vol. 6, pp. 2935-2939. IEEE, 2010.

[3]. Feng, Xinyu. "Design and analysis of mobile agent communication protocols." PhD diss., Nanjing University, China, 2002.

[4]. Anne Nguyen, Ian Stewart, Xinfeng Yang, "A mobile Agent: Applications for E-Commerce", AusWeb01, the Seventh Australian World Wide Web Conference, 21st-25th April, Opal Cove Resort, Coffs Harbour, NSW.© 2000.

[5]. Tina Setter, Andrea Gasparri, and Magnus Egerstedt. "Trust-based interactions in teams of mobile agents." In 2016 American Control Conference (ACC), pp. 6158-6163. DOI**:** 10.1109/ACC.2016.7526637, IEEE, 2016.

[6]. Satoh Ichiro, "Building reusable mobile agents for network management." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 33.3 (2003): 350-357.

[7]. Schoeman, Marthie, and Elsabé Cloete. "Architectural components for the efficient design of mobile agent systems." Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology. South African Institute for Computer Scientists and Information Technologists, 2003.

[8]. Lange, Danny B., Mitsuru Oshima, Günter Karjoth, and Kazuya Kosaka. "Aglets: Programming mobile agents in Java." In Worldwide Computing and Its Applications, pp. 253-266. Springer Berlin Heidelberg, 1997.Robert Gray, David Kotz, Saurab Nog, Daniela Rus, George Cybenko, "Moblie Agents: The Next Generation in Distributed Computing", Deptt. Of CSE, Dartmouth College, 1997, IEEE.

[9]. Ahila, S. Sobitha, and K. L. Shunmuganathan. "Overview of mobile agent security issues—Solutions." In Information Communication and Embedded Systems (ICICES), 2014 International Conference on, pp. 1-6. IEEE, 2014.

[10]. Rajesh Kumar, S Niranjanr, and Yashpal Singh, "A Review on Mobile Agent Technology and Its Perspectives." Journal of Computer Sciences and Applications, vol. 3, no. 6 (2015): 166-171. DOI: 10.12691/jcsa-3-6-11.

[11]. Pandey, Mr Rajesh, Mr Nidheesh Sharma, and Mr Ramratan Rathore. "Aglets (A Java Based Mobile Agent) And Its Security Issue." International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)2.4 (2013).

[12]. Gray, Robert, David Kotz, Saurab Nog, Daniela Rus, and George Cybenko. "Mobile agents: The next generation in distributed computing." In Parallel Algorithms/Architecture Synthesis, 1997. Proceedings. Second Aizu International Symposium, pp. 8-24. IEEE, 1997.

[13]. Lange, Danny B., Mitsuru Oshima, Günter Karjoth, and Kazuya Kosaka. "Aglets: Programming mobile agents in Java." In Worldwide Computing and Its Applications, pp. 253-266. Springer Berlin Heidelberg, 1997.

[14]. Mitsuru Oshima, Guenter Karjoth, "Aglets Specification", Departamento de Lenguajes y Sistemas Informáticos Universidad de Sevilla, Copyright © 1997, 1998 IBM Corp.

[15]. Mitsuru Oshima, Guenter Karjoth, "Aglets Specification", Departamento de Lenguajes y Sistemas Informáticos Universidad de Sevilla, Copyright © 1997, 1998 IBM Corp.

[16]. Shigeki Shiokawa, "Performance analysis for use of mobile agent in wireless multihop networks." In 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 827-832. DOI**:** 10.1109/ICUFN.2016.7537153, IEEE, 2016.

[17]. Lange, Danny B. "Mobile objects and mobile agents: the future of distributed computing?." In ECOOP'98—Object-Oriented Programming, pp. 1-12. Springer Berlin Heidelberg, 1998.

[18]. S. Sobitha Ahila, and K. L. Shunmuganathan., "Overview of mobile agent security issues—Solutions." In Information Communication and Embedded Systems (ICICES), 2014 International Conference on, pp. 1-6. IEEE, 2014.

[19]. Ma, Lu, and Jeffrey JP Tsai. "Security modeling and analysis of mobile agent systems" Vol. 5. World Scientific, 2006.

[20]. Li, Wei, and Minjie Zhang. "MAT: a mobile agent system for supporting autonomous mobile agents." Journal of Research and Practice in Information Technology 33.3 (2001): 211-227.

[21]. Li, Wei, and Minjie Zhang. "Distributed Task Plan: A Model for Designing Autonomous Mobile Agents." In IC-AI, pp. 336-342. 1999.

[22]. H. M. Eldegwi, M. B. Badawy, and Hamdy M. Kelash., "Building a Secure Decentralized Energy System with Remote Monitoring Using Mobile Agents." In 2015 Fifth International Conference on e-Learning (econf), pp. 263-268. DOI**:** 10.1109/ECONF.2015.80, IEEE, 2015.