International Research Journal of Computer Science (IRJCS)
Issue 10, Volume 4 (October 2017)
ISSN: 2393-9842
www.irjcs.com

A STANDARD FRAMEWORK AND MIGRATION PROCESS OF MOBILE AGENTS USING PRE-PROCESSING TECHNIQUES

L. Kathirvel kumaran
Research Scholar, Department of Computer Science
Rathinam College of Arts and Science, Coimbatore, Tamil Nadu, INDIA

R. Muralidharan,
Assistant Professor & Head, Department of Computer Science
Rathinam College of Arts and Science, Coimbatore, Tamil Nadu, INDIA

Manuscript History
Number: **IRJCS/RS/Vol.04/Issue10/OCCS10082**
DOI: 10.26562/IRJCS.2017.OCCS10082
Received: 08, September 2017
Final Correction: 23, September 2017
Final Accepted: 02, October 2017
Published: October 2017
Citation: L.Kumaran, and R. Muralidharan. research report, Issue X. Rathinam College of Arts and Science, Coimbatore, Tamil Nadu, INDIA, (October 2017).IRJCS
Editor: Dr.A.Arul L.S, Chief Editor, IRJCS, AM Publications, India

Abstract-- A Mobile agent is "a program that is self-governing enough to act separately, even when the user or application that launched it is not available to provide guidance and handle errors". In general terms, it is a program that acts in behalf of its owner. A mobile agent is an object that migrates through many nodes of a assorted network of computers, under its own control, in order to perform tasks using resources of these nodes. The mobility trait of a mobile agent implies operation thereof in untrustworthy environments, which introduces malicious host threats. Available literature have been studied, analyzed and discussed. The salient characteristics as well as the drawbacks of current solutions were isolated. Through this knowledge a dynamic mobile agent security framework was defined. The framework is based on the definition of multiple security levels, depending on type of deployment environment and type of application. A prototype was constructed and tested and it was found to be lightweight and efficient, giving developers. Insight into possible security threats as well as tools for maximum protection against malicious hosts. The framework outperformed other frameworks / models as it provides dynamic solutions without burdening a system with unnecessary security gadgets and hence paying for it in system cost and performance.

Key terms: Mobile agent security, malicious hosts, Security framework requirements, Mobile agent security framework, Countermeasure classification, Communication networks, Computer security Mobile Agents, E-commerce, M-commerce, Active object, Node.

I. INTRODUCTION

The client-server paradigm is introduced with the concept of remote procedure calls as the basis for distributed computing. In traditional approaches such as these, distributed information access brought the data to the point of computation.

This however, required crucial conditions such as a continuous link between the client and the server when requesting information. The next step from the client-server technology is the introduction of mobile agents. The concept of a mobile agent brings the computation of data and its mobility and autonomy attributes make permanent connections unnecessary. The notion of a mobile agent arose from that of a software agent. Software agent can be defined as software components that communicate by exchanging messages in an agent communication language [1]. The concept of software agents is based on objects as recognized in the object-oriented environment and they can adopt several different forms such as stationary agents, intelligent agents and mobile agents. A mobile agent can be defined as an autonomous program that moves between networks to take advantage of the services supplied by stationary agents [2]. The stationary agent resides on a specific host with the inability to move about but with the ability to offer services or perform tasks on behalf of its owner. A mobile agent, on the other hand carries along its complete implementation and interacts with a host system as well as other mobile and stationary agents. Mobile agents have a distinct computational advantage by moving computation close to the resources they need to access, hence reducing network communication, bandwidth and latency. Mobile agents are deployed for various purposes like information searching, filtering and retrieving applications, low level network maintenance, testing, fault diagnosis and the dynamic upgrading of existing services, concluding certain e-commerce deals or negotiating with other mobile or stationary agents [3]. An agent is generally regarded as mobile when its execution can be interrupted (usually briefly), before it migrates to a new host and is then resumed after the transportation to the new runtime environment. Mobile agents do not transport themselves, but depend on the mobile agent system to move their binary images between execution layers over a variety of media.

Mobile agents open several new possibilities for conducting business in a network and especially the Internet environment, but they also introduce a new dimension of security issues. In fact, full-scale adoption of mobile agent technology in untrustworthy network environments, such as the Internet, has been hampered and delayed by several security complexities. Mobile agent categorizes threats in the mobile agent environment into four distinct classes, namely threats imposed by the mobile agent to the host, threats imposed by a mobile agent host to a mobile agent, threats imposed by a mobile agent to another mobile agent and threats imposed from other entities to mobile agents. A Mobile agent is "a program that is sovereign enough to act in antagonism, even when the user or application that launched it is not available to provide regulation and handle errors". In general terms, it is a program that acts in behalf of its holder. A mobile agent is an object that migrates through many nodes of a assorted network of computers, under its own control, in order to perform tasks using resources of these nodes. A Mobile Agent is a type of software agent with the feature of autonomy, social ability, learning, and most importantly. The mobile agent is a process that can transport its state from one environment to another, with its data intact, and be capable of performing appropriately in the new environment. A mobile agent is a precise form of mobile code, within the field of code mobility. However, in contrast to the isolated valuation and Code on demand programming paradigms, mobile agents are dynamic in that they can desire to drift between computers at any time during their implementation [4]. This makes them a powerful tool for implementing distributed applications in a computer network. An untie multi agent system is a system in which agents that are owned by a mixture of stakeholders incessantly enter and disappear the system. A mobile agent is an object that migrates through many nodes of an assorted network of computers, under its personal control, in order to perform tasks using resources of these nodes. The uses of this technology represent a change in the disseminated programming paradigm [5]. This approach provides many benefits to the development of distributed applications but introduce new necessities to the engineering of these systems. The development of distributed applications is directly influenced by the choice of an architecture style. The necessities of the system as scalability, fault tolerance, response time, and support for disconnected operations and so on, are important point to be measured and reasoned before the implementation of a system.

II. ADVANTAGES AND USES OF MOBILE AGENTS

The mobile agent paradigm introduces several advantages. Some of the most salient include:

- The mobile agent is not bound to the system where it begins execution. A mobile agent 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 [9].
- Both bandwidth limitations as well as the support for disconnected operation capabilities are eminent problems experienced in the wireless and mobile environments. By moving the computation to the host and as a result decreasing the amount of packets on the network, mobile agents can assist in alleviating these problems [10].
- Mobile agents provide the ability to conduct intelligent information retrieval, such as retrieving appropriate information from a number of hosts as well as performing some computations.

- Mobile agents overcome network latency in that real-time systems need to respond to changes in their environment. Controlling such systems through a large network involves significant latencies for which mobile agents can offer a solution.

These advantages open up several applications that will benefit from the use of mobile agent systems. Typical applications range from information searching, filtering and retrieval to electronic commerce on the Web where they act as personal assistants for their owners. Mobile agents can also be used in network management maintenance, testing, fault diagnosis, and for dynamically upgrading the capabilities of existing services .Other uses include workflow management, air traffic control, information retrieval management and education.

III. NATURE OF MOBILE AGENT

A mobile agent consists of the program code and the program execution state. Originally a mobile agent resides on a computer called the abode machine. The agent is then dispatched to perform on an isolated computer called a mobile agent host (a mobile agent host is also called mobile agent platform or mobile agent server). When a mobile agent is dispatched the complete code of the mobile agent and the execution state of the mobile agent is transfer to the host. The host provides a appropriate execution environment for the mobile agent to execute [7]. The mobile agent uses wherewithal (CPU, memory, etc.) of the host to perform its task. After completing its task on the host, the mobile agents migrate to another computer. While the state information is also transferred to the host, mobile agents can resume the carrying out of the code from where they left off in the previous host instead of having to restart execution from the commencement. This continues awaiting the mobile agent returns to its home machine after completing execution on the last machine in its schedule.

IV. MOBILE AGENTS FUNCTIONALITY

Mobile agents are distinct as active objects or cluster of objects that have performance, state and position.

- Mobility: Agents that can travel in network
- Autonomy: Agent itself decides when and where to migrate next

Mobile Agent travels from node to node of a distributed system performing tasks in behalf of its owner. At the end of this process, an agent can return to its abode site and report itself to the users who inject this object in the disseminated system. Mobile agents decide when and where to move [4]. Movement is often evolved from Remote Procedure Call (RPC) methods. As like a user directs an Internet browser to "visit" a website, a mobile agent accomplishes a move through data duplication. As the interaction between the agent and the resource after moving is perform in the similar host, not including the transmission of messages through the network, this paradigm is indicate for some kinds of real time distributed applications[5].

V. LIFE CYCLE OF MOBILE AGENT

- The mobile agent is created in the Client Machine.
- The mobile agent is dispatched to the Server A for execution.
- The agent executes on Server A.
- After execution the agent is cloned to create two copies. One copy is dispatched to Server B and the other are dispatched to Server C.
- The cloned copies execute on their respective hosts.
- After execution, Server B and Server C send the mobile agent received by them back to the Client Machine.
- The Client Machine retracts the agents and the data brought by the agents is analyzed. The agents are then disposed.

VI. MOBILE CODE SECURITY FRAMEWORKS

TAN [11] described a method by which execution traces are enhanced through a trusted third party called a verification server. The construction of a mobile agent is simplified by making use of mobile agent templates. The execution tracing protocol as proposed by Vigna (1998) [12] is changed in this framework through the introduction of a verification server that is responsible for verifying the traces, instead of the local host. The framework consists of a certification authority, responsible for the issuing of certificates to other entities in the framework as well as the management of keys and a verification server, which is a trusted third party responsible for the verification of execution traces submitted by hosts on behalf of the agent owner. Two types of certificates are used, namely capability certificates and execution certificates. Capability certificates associate the identity of the host with its capability of correctly executing the mobile agent template. A mobile agent template identifier replaces the public key present in a normal certificate. The private key of the verification server signs these certificates. Execution certificates identify the success of the validation process and are generated by the verification server for a host. An execution certificate contains a hash of the agent's code and state, a timestamp, identity of the verification server, identity of the host and the results of the trace.

A record of all invalid execution traces that were detected is kept in a capability certificate revocation list. The verification server submits an entry containing the identity of the server, identity of the host platform, fault detected in the trace and a timestamp. Before migrating to a new host, the following occurs: The mobile agent contains a list of template identifiers that represents the templates it is composed from. This as well as the code of the mobile agent is signed by the agent owner platform. The identifiers are sent to the new host platform, which checks if it possesses capability certificates containing some or all identifiers specified. The capability certificates are sent back to the mobile agent for review in order to decide to migrate or not [13]. After migration to the new host platform, the mobile agent is executed and an execution trace is created by the platform. The created trace is submitted to the verification server where it is validated and an execution certificate is prepared and sent back to the host platform (once the trace is validated). The host platform keeps a copy of the execution certificate and the original is sent with the mobile agent to the next host platform. If the trace is not validated no certificate is issued and an entry is written to the capability certificate revocation list. The Mobile code security framework is advancing on the current execution traces technique through the use of a trusted certification authority. This trusted entity is not only responsible for key management but also the verification of the traces and the subsequent issuing of certificates upon validation. This framework can effectively be implemented in environments where such certification authorities exist. Possible network congestion at the verification server can be a problem if the number of hosts and mobile agents compared to the number of verification servers are high. Execution tracing as a countermeasure also causes extensive extra overhead in terms of computational resources as well as additional communication sessions.

VII. METHODOLOGIES

After an agent has sensed its environment, it needs to form an internal representation. Furthermore, based on this representation, the agent selects which action to perform, i.e. how to react to the state of the environment. However, as in context-aware applications, the sensor measurements often contain errors and some measurements might be missing, the raw signals often need to be preprocessed before reliable inferences can be made [11]. In order to implement cost minimized search, parallel searching is used [5]. In this technique, multi - mobile agents are used to retrieve information from different servers in parallel and the response will be sent back to the requested user.

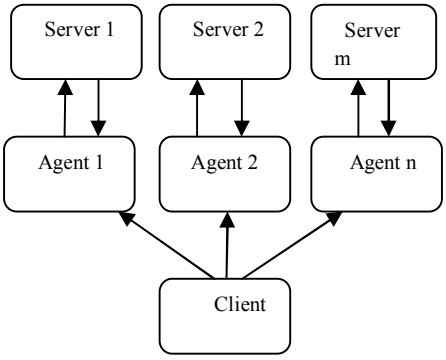

Fig. 7.1 Parallel Searching - Agents

Fig 7.1 illustrates that the search query requested through client will be send to multiple mobile agents which will parallel search from different servers and send the result back to the client and hence perform cost minimized search.

Parser. In the first phase, the query is parsed and translated into an internal representation that can be easily processed by the later phases. The same parser can be used for a centralized and distributed database system.

Query Rewrite. Query rewrite transforms a query in order to carry out optimizations that are good regardless of the physical state of the system (example, the size of tables, presence of indices, locations of copies of tables, speed of machines, etc.). Typical transformations are the elimination of redundant predicates, simplification of expressions, and nesting of sub queries and views.

Query Optimizer. This component carries out optimizations that depend on the physical state of the system. The optimizer decides which indices to use to execute a query, which methods (example, hashing or sorting) to use to execute the operations of a query (example, join and group-by), and in which order to execute the operations of a query. The query optimizer also decides how much main memory to allocate for the execution of each operation.

Plan. A plan specifies precisely how the query is to be executed. Probably every database system represents plans in the same way: as trees. The nodes of a plan are operators, and every operator carries out one particular operation (example, join, group by, sort, scan, etc.).

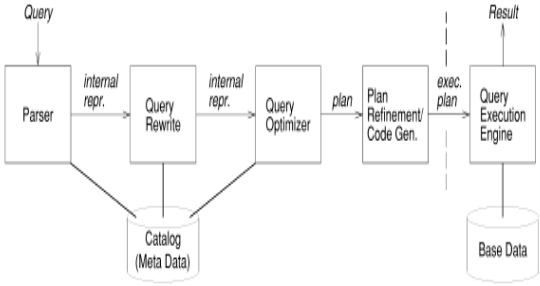

Fig. 7.2 Phases of Query Processing

Plan Refinement/Code Generation. This component transforms the plan produced by the optimizer into an executable plan. In some systems, plan refinement also involves carrying out simple optimizations which are not carried out by the query optimizer in order to simplify the implementation of the query optimizer.

Query Execution Engine. This component provides generic implementations for every operator. All state-of-the-art query execution engines are based on an iterator's model. In such a model, operators are implemented as iterator's and all iterator's have the same interface. As a result, any two iterator's' can be plugged together, and thus, any plan can be executed.

Catalog. The catalog stores all the information needed in order to parse, rewrite, and optimize a query. It maintains the schema of the database (i.e., definitions of tables, views, user-defined types and functions, etc.). It should be noted that the architecture shown in Fig. 7.2 and described in this subsection is not the only possible way to process queries. There is no such thing as a perfect query processor. An alternative architecture has, for example, been developed. In that architecture, query rewrite and query optimization are carried out in one phase. Furthermore, there have been proposals to optimize a set of queries rather than individual queries.

VIII. ARCHITECTURES IN STATE-OF-THE-ART MOBILE AGENT PLATFORMS

Several research laboratories and industrial manufacturers were involved in the development of various MA platforms since 1996. Those platforms were built on top of different operating systems, and based on different programming languages and technologies. Even new languages have been realized, exclusively designed for the support of mobile agents (e.g., Tele Script). However, what is more relevant from the middleware perspective is that common trends in MA platforms have started to emerge evidently within the last couple of years. Interpreter-based programming languages, particularly Java, are forming the basis for most of today's agent platforms, mainly due to their easy portability over heterogeneous platforms. In addition, Java is frequently chosen for the available support, directly at the language level, of fine-grained security policies and transport facilities via object serialization [Gosling, 97]. Moreover, even if coming from different experiences and domain-specific design constraints, the implementers of MA platforms are achieving a general agreement on the architecture of middleware services that are necessary for supporting mobile agents in open and global environments and for leveraging the diffusion of MA-based services in the Internet. Finally, several approaches have recently explored the possibility of integrating mobile agents and RPC based middleware like the OMG Common Object Request Broker Architecture (CORBA), possibly stimulated by the research work accomplished for the definition of agent interoperability standards. Good examples of state-of-the-art MA systems are Aglets Workbench from IBM Japan , Concordia from Mitsubishi, Odyssey from General Magic, Voyager from Object Space, Ajanta from the University of Minnesota, TACOMA from Universities of Cornell and Tromso , Grasshopper from IKV GmbH [Grasshopper], and SOMA from University of Bologna [SOMA]. An extensive description and comparison of MA platforms is out of the scope of this that can be found in MA platforms typically realize a distributed processing environment, consisting of several middleware components, and usually referred to as Distributed Agent Environment (DAE). DAEs usually support a hierarchy of locality abstractions (domains, agent systems and places) to model network physical resources, and two different types of agents (mobile and stationary). Figure 8.1 depicts an abstract view of these entities. The actual runtime environment for agents is called agent system: on each network host, at least one agent system has to run to support the execution of agents. Agent systems are structured by means of places, i.e., isolated execution contexts that are able to offer specific additional services. For example, there may exist a communication place offering sophisticated communication features, or there may be a trading place where agents offer/buy information and service access. Agent systems can be grouped into domains that usually model a physical local area network: each domain has an associated (domain) registry that maintains information about all registered agent systems, places and agents. The domain locality abstraction facilitates the management of the distributed components (agent systems, places and agents) in the DAE and improves its scalability.

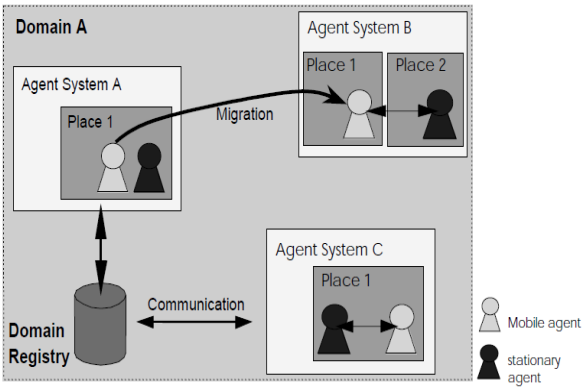

Fig 8.1 Structure of a Distributed Mobile Agent Environment (DAE)

While agent systems and their places are generally associated to a single domain for their entire lifetime, mobile agents are able to move between different agent systems of possibly different domains. The current location of mobile agents is updated in the corresponding domain registry after each migration. By contacting the domain registry, other entities (e.g., agents or human users) are able to locate agents, places, and agent systems residing in a domain. Two different types of agents are defined. Mobile agents are able to move from one physical network location (agent system A in Figure 8.1) to another one (agent system B). Stationary agents, instead, are bound to the agent system where they have been installed and where they remain for their whole lifetime to provide a place persistent service according to the Client Server model of interaction. Several fundamental requirements have been identified due to the experiences made during research and development activities in the MA area. These requirements are fulfilled by any state-of-the-art MA platform, and their identification is the first fundamental step towards the definition of a common and interoperable distributed middleware to support mobile agents in the Internet scenario. The implementation of a modern MA system requires middleware components to support:

- Agent execution. An MA platform must provide the basic capability to put incoming mobile agents into execution, taking into account possible agent specific requirements regarding the runtime environment (e.g., binding to specified local resources). The platform has to retrieve the agent code that may be either delivered with the migration request or downloaded separately from an external code base.
- Transport. A special mobility support must be provided by the platform, not only to facilitate the network transport of agent code and execution state, but also to permit MA system administrators to command remote execution and migration. Note that both agent execution and transport cannot be sufficiently handled without a strict interworking with the security support mentioned in the following.
- Unique identification. Mobile agents as well as agent systems have to be uniquely identifiable in the scope of the entire Internet environment. Thus, special support is required for the generation of unique agent and agent system identifiers.
- Communication. Agents should be able to communicate with each other as well as with platform services. Several mechanisms are possible, such as messages, method invocation, object sharing and tuple-spaces, with different levels of expressive power and of temporal/spatial coupling between coordinating entities. Communication through messages may be done point-to-point, by multicasting, or by broadcasting.
- Security. Basic issues are authentication (i.e., the determination of the identity of an agent or an agent system), and access control of resources/services depending on the authenticated identity of the requesting entity. To guarantee privacy and integrity, crucial information such as code and state of a migrating agent should exploit public-key cryptographic encryption before transfer over an untrusted network.
- Management. It is necessary for agent administrators to be able to remotely monitor and control mobile agents and MA-based provided services. Control functions include temporary interruption of the execution of an agent task, agent premature termination, and modification of its task list. The monitoring of an agent is associated with its localization in the scope of the whole distributed environment. Regarding an agent system, all hosted agents as well as the occupied system resources have to be monitored and controlled, possibly to notice and avoid denial-of-service attacks. Figure 8.2 shows the structure of a core agent system that includes several services in order to fulfill the basic functional requirements identified above.
- Note that some services provide remote interfaces in order to be accessible by external actors, such as other agent systems, agents, or human users.

Apart from the basic capabilities shown in the figure, additional ones have been taken into consideration in some of the most recent MA platforms. For instance, an interoperability module is offered to permit the integration of heterogeneous agents and agent systems with already existing services and legacy components.

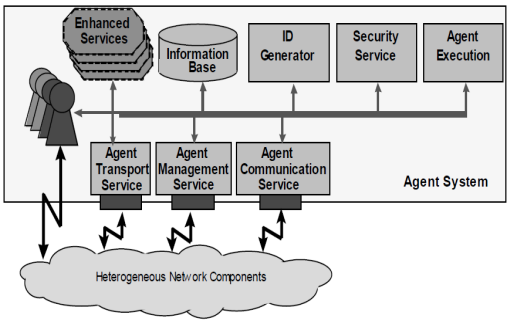

Fig 8.2 Architecture of basic facilities in mobile agent platforms

Interoperability is obtained via compliance with emerging standards in the MA area, all based upon the CORBA middleware, and the implementation of the interoperability facility in SOMA .Other capabilities have started to be accepted as fundamental and tend to be integrated in MA platforms. The persistency facility, for example, permits to temporarily suspend executing agents and to store them on a persistent medium. Persistency allows agents not to waste system resources while they are waiting for external events such as the reconnection of one user or terminal where they have to yield back the results of their operations. In addition, a facility for the mobile computing support is provided in some MA systems to accommodate the nomadicity of users, terminals and service components, which can move with no need to suspend offered/accessed services. These additional features can significantly benefit from the modular organization of MA platforms and should be handled as add-ons that can be "plugged" into a core MA system to dynamically extend its basic functionality.

IX. SECURITY LEVELS

As different types of applications call for diverse security mechanisms, it is vital that the mobile agent developers are enabled with tools to build more secure mobile agent applications. Distinguishing between types of applications and deployment environments is at the base of the proposed framework. it allows the agent developer to do a proper evaluation of the potential threats of a specific deployment environment; it allows the agent developer to select only specific and necessary countermeasures that could defend against the potential threats; it allows for the construction of a more light-weight and secure mobile agent application that target only threats that are a reality instead of carrying along superfluous countermeasures; considering the above advantages, it allows for a more cost effective, yet protected mobile agent system; once again, by considering the above advantages, it allows for the construction of an application with improved performance, where no unnecessary computations are conducted. A large number of current mobile agent security frameworks are designed and developed for specific mobile agent applications (for example Electronic supermarkets [6] As a result specific designs can often not easily be reused, extended or transferred to other, different applications. This is where the proposed framework is significant, being both dynamic (adaptable) and not application specific. Even though, existing systems often use security measures inherent to the underlying operating system or virtual machine, the proposed framework offers the opportunity to add additional (needed) countermeasures based on environmental and application evaluation without redesigning the entire application.

X. RESULTS AND DISCUSSION

The aim of performance evaluation is to evaluate the effectiveness of each of the proposed techniques described in this paper. The analytical models presented in the previous section parallel analyze the elements of each processing component. These models are then incorporated into a simulation model, and the results of our simulation experimentations are presented in the following sections. The grouped results based on the three key contributions in this paper, which include Multithreaded Query Execution, Top n and Bottom n queries, trusted platforms and it is compared with Mobile Agent Preprocessing. In the simulation, the analytical models presented earlier are incorporated, to simulate the record and processing distribution. In the experimentations, it is particularly focus on process time, number of packets to be sent and number of queries to be sent. The results are presented as Fig. 10.1. Which depicts the comparison chart of the four techniques that is used in the thesis, where Multithreaded Query Execution, Top n and Bottom n queries, trusted platforms provides a lower percentage of result when compared to the Mobile Agent Preprocessing.

There are three constraints that are measured with the same number of queries. They are:

1. Query process time
2. Query Response time
3. Number of packets

All the three parameters produced a positive result, when compared with the existing techniques. From the table 10.1, the number of queries sent to the server is kept fixed, so that this parameter is used to measure all the remaining above three parameters.

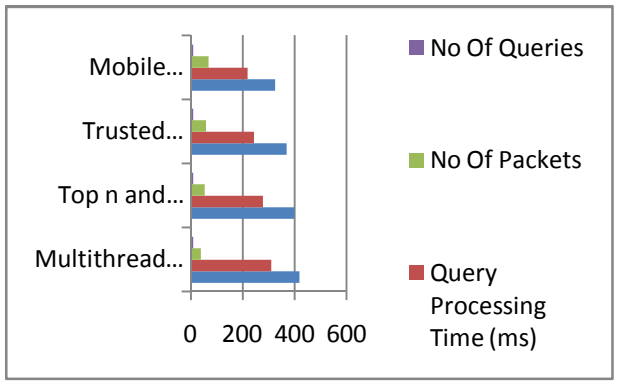

Fig. 10.1 Comparison of four methods Graphical Pattern

The query process time for the multi-threaded query execution is 420ms whereas the top n and bottom n queries are 400ms which is considerably reduced in the mobile agent preprocessing which is 325ms. But it in individual trusted platform technique, it is 370ms. Another measure is query response time, where the query taken to respond to the central server after retrieving the data from the database, which will varies from the trusted platform to that of the mobile agent preprocessing. Similarly, the number of packets to be sent also varies from query execution techniques to mobile agent preprocessing.

TABLE.10.1 COMPARISON OF BANDWIDTH WITH AND WITHOUT QUERY PROCESSING OF MOBILE AGENTS FOR A FIXED NUMBER OF PACKETS

Methods	Multithread query Execution	Top n and Bottom n Queries	Trusted platform	Mobile Agents with Preprocessing
Query Process Time (ms)	420	400	370	325
Query Processing Time (ms)	310	280	245	220
No Of Packets	40	55	60	70
No Of Queries	10	10	10	10

TABLE 10.2 COMPARISONS OF ALL THE METHODS

No of Packets	Bandwidth consumed for Mobile Agent without Preprocessing (kbps)	Bandwidth consumed for Mobile Agent with Preprocessing (kbps)
10	12	10
10	10	8
10	8	7
10	9	5

From fig.10.2, comparison of bandwidth for mobile agent with and without query processing is done for a fixed number of packets that is to be sent from sender to the receiver. From Table 10.2, the packets that is to be sent is 10 and the bandwidth differs for all the techniques and the mobile agent preprocessing consumes a low bandwidth of 5 kbps when compared to all the above techniques, the main advantage is that, the mobile agent server stores the retrieved data from the concerned servers for a particular TTL, so that it need not fetch the particular data once again, while the request is being generated for the second time.

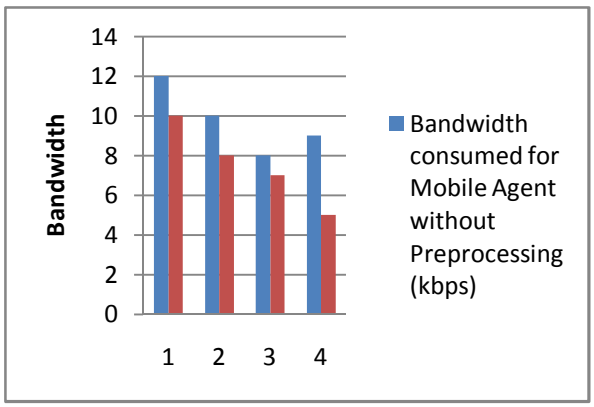

Fig. 10.2 Comparison of Bandwidth

XI. CONCLUSIONS AND FUTURE ENHANCEMENT

The Mobile agents are secluded from routing to malicious host. Also it will help to retrieved data so far will be returned back to the requesting client once the malevolent host is detected. With several advances, the mobile agents will be an important ingredient in producing secure, flexible distributed systems. Also it is focused on a specific part of the overall architecture, which supports distributed preprocessing in ubiquitous environments. In addition, it also included details about the sensing mechanisms of the agents. The framework outperformed other frameworks / models as it provides dynamic solutions without burdening a system with unnecessary security gadgets and hence paying for it in system cost and performance. The use of mobile agent can lead to huge communication savings. As a future work, this model can be further extended for dynamic query updating on the mobile agent query processing server and online purchasing system can be developed on a single web page, with security concern and we will planning to use mobile agent to answer semantic queries in mobile scenarios. In addition, it is also necessary in order to find out how well economic models and data dissemination models work for large-scale query processing.

REFERENCES

1. Genesereth, m.r. & ketchpel, s.p. 1994. Software agents. *Communications of Acm*, 37:48-53.
2. Karnik, n. 2000. Security in mobile agent systems. *Technical report*. University of Minnesota.
3. Sakaguchi et al Christoffel M., Pulkowski S., Schmitt B., Lockemann P.C. (2005) Electronic Market: The Roadmap for University Libraries and Members to Survive in the Information Jungle. SIGMOD RECORD, 27(4), 68-73.
4. Lesser R. S. Silva Filho, J. Wainer, E. R. M. Ma-deira, C. Ellis – CORBA Based Architecture for Large Scale Workflow. Special Issue on Autono-mous Decentralized Systems of the IEICE Transac-tions on Communications, Tokyo, Japan, Vol. E83-B, No. 5. May 2009, pp.988-998.
5. Quinn, A., Tesar, L.: A survey of techniques for preprocessing in high dimensional data clustering. In: Proceedings of the Cybernetic and Informatics Eurodays.(2000)
6. L.Kathirvelkumaran: "Data Communication through Mobile Agent with Pre-Processing Techniques in Electronic Environment ".Advance in Electronic and Electric Engineering. ISSN 2231-1297, Volume 3, Number 8 (2013), pp. 983-986
7. Maw Min1, and NyeinNyein Mobile Agent-based Information Retrieval for Shopping Assistant. Proceedings of 2015 International Conference on Future Computational Technologies (ICFCT'2015).
8. Anthony H. W. Chan, Caris K. M. Wong, T. Y. Wong, and Michael R. Lyu.,2012., Department of Computer Science and Engineering , The Chinese University of Hong Kong, Shatin, N. T., Hong Kong "Design, Implementation, and Experimentation on Mobile Agent Security for Electronic Commerce Applications."
9. LANGE, D.B. 1998. Mobile objects and mobile agents: the future of distributed computing. In: Proceedings of the European Conference on Object Oriented Programming (ECOOP'98). Brussels, Belgium, Invited Talk.
10. Suri, N., Bradshaw,J.M Breedy, 2000. An overview of the NOMADS mobile agent system. In: ciar & bryce (eds.). 6th ECOOP workshop on mobile object systems. France.
11. L.Kathirvelkumaran, R.Muralidharan A Migration Process of Mobile Agent using Pre processing Technique Proceedings. 5th international conference on contemporary issue in agricultural, Engineering, Management, Information Technology and Life Science, May 2017,Kandy,Srilanka.
12. TAN, H.K. & MOREAU, L. 2002. Certificates for Mobile Code Security. In:Proceedings of the 17th ACM Symposium on Applied Computing (SAC'2002):76.
13. L.Kathirvelkumaran, R.Muralidharan A STANDARD EVACUATION PROCESS OF MOBILE AGENTS USING PRE-PROCESSING TECHNIQUES. International journal of research in computer applications and Management Volume 7 June 2017 Issue No 6.
14. VIGNA, G., CASSELL, B. & FAYRAM, D. 2002. An Intrusion Detection System for Aglets. In: Suri, N (ed.). Proceedings of the International Conference on Mobile Agents. Barcelona, Spain.

