ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/107978, December 2016* ISSN (Online) : 0974-5645

An Effective Load Balancing through Mobile Agents using Aglet System

G. Ravindran* and S. Kamakshi

School of Computing, SASTRA University, Thirumalaisamudram, Thanjavur - 613401Tamilnadu, India; ravindran.lgr@gmail.com,kamakshi@cse.sastra.edu

Abstract

Background/Objectives: Security is an important feature for the communication of data in the mobile network. But, however the existing work does not satisfy the security feature of the communication of data in the network. Methods: Due to lack of security in mobile agents, there are many chances of security breach through intruders. To overcome these disadvantages, a general secured aglets system has been proposed and load balancing is achieved effectively through various models. **Findings:** The evolved aglet system exhibits high security under the external attackers. Hence, it distributes workload for all intermediate client users and determines various time calculations like Total run-time and Elapsed-time. **Applications:** Some mobile agent applications are Mobile Computing, Personal Assistant, E-commerce, Data Collection, Secure Brokering, Information Distribution, Network Security testing, Database searches and Parallel Computing.

Keywords: Aglets, Load Balancing, Mobile Agents, Security

1. Introduction

1.1 Mobile Agents

Mobile agents[1] are independent programs that can be drifted from one host to another host in a heterogeneous network. In mobile agents, functions can be obtained effectively. It is also used to implement applications for the distributed systems with robust framework. Besides this, it offers betterment to the latency, bandwidth of master-slave applications and shrinking vulnerability to disconnecting the network. Mobile agents[2] have several advantages and disadvantages in the development of different services in the application of distributed system. The benefits of using mobile agents are to reduce network load, to design robust and fault–tolerant systems, to overcome the network latency and so on. The mobile agent examples[2] are Aglet, Concordia, voyager, agent TCL, which are mainly used for the safety of the mobile codes.

1.2 Aglets

IBM Aglets Workbench (Aglets) was developed by in from IBM Tokyo Research laboratory in 1996. Aglet[3] is a Java object that is able to migrate from a host to another host in a network. Aglets working in a host could stop its execution, migrate to other host, and continue its execution. When aglet migrates, it carries the program code and each state from every single object carrying it. A built in security mechanism is able to secure a host from an un-trusted aglet. The word "Aglets" is the combination of agent and applet. The difference with an applet is that aglet also carries state and itinerary (migrating plan). Aglets Software Development Kit (ASDK) is a software packages which can be used to write a mobile agent application. ASDK is based on Java language and can be obtained freely from the internet either in source code form or byte code from compiled binary file.

1.3 Aglet API

An Aglet interacts with its environment through an Aglet Context object[4]. Aglets are always executed in Aglet Contexts. It interacts with one to other; Aglets goes to Aglet-Proxy objects. The Aglet-Proxy object role as an interface of an Aglet and accommodate generally Aglet access behind the code. In this pathan Aglet-Proxy object can protect an agent against duplicate agents.

* *Author for correspondence*

1.4 Aglet Communication Model

The key components of aglet communication model are:

- ***Message:*** A message is treated as an object in aglets. It is exchanged between aglets, which allow synchronous and asynchronous message passing systems. It is also used to combine and transfer information in a loosely coupled in nature.
- ***Future Reply:*** Asynchronous message uses this future reply for sending handle so that the sender can receive an acknowledgement asynchronously.
- ***Reply Set:*** A reply set is a set which holds various future replies. By using this, the results can be obtained.

1.5 Basic Operations

Some of the basic operations involved in an Aglet are: Creation, Cloning, Dispatching, Retraction, Deactivation, Activation, and Disposal.

- ***Creation***: In this context, aglet can be created and its identifier will be assigned for the newly created aglet. After the aglet is inserted into the context, its initialization process is started.
- ***Cloning***: The cloning process produces duplicate image of the original aglet in the same context. The duplicated image differs from the original aglet by their assigned identifier.
- ***Dispatching***: An aglet is dispatched from one agent to another agent by removing all aglet functionalities from its current agent and transferring the functionalities to the destination agent where the functionalities need to be executed.
- ***Retraction***: In Aglet retraction process, the aglet context is removed from its current agent and transfers the aglet context to the requested agent.
- ***Deactivation:*** Deactivation is the process of halting the execution of an aglet and saves its data into secondary storage.
- ***Activation***: Activation is the process of restoring the existing state from the secondary storage.
- ***Disposal***: Disposal is the process of halting the execution and disposing the context from the secondary database.

1.6 Aglet Environment

Tahiti is an application program serving the work as aglets server. Tahiti is bundled inside one package with ASDK. A number of servers (Tahiti) can run in a single computer by labeling each one of them with different port number. Tahiti also provides user interface for monitoring, creating, sending, and extermination of an aglet and also determining privilege access for an agent.

In has introduced the Threshold algorithm[5] which is used for assigning the loads during the node creation. Remote message is sent without selecting a node. Every node has one copy of private load systems. Categories the load in some levels: Under-Load, Over-Load and Medium and parameters like Xmin and Xmax of these levels.

Under-Load →load <Xmin
Medium →Xmin<= load< = Xmax
Over-Load→load >Xmax

At the beginning, every node will be treated as under loaded level. When the work load of a node reaches its maximum limit, then it forwards the messages concerning the new work load state to all host nodes, on a regular basis modernizing them as to the existent work load state. If the state of local is not overloaded, then the respective work load is apportioned locally. In other respects a host node under loaded node is elected and if no such nodes survive it is also apportioned locally. In a Threshold algorithm has low inter procedure information and high number of local process allotment. It will reduces the overhead of remote procedure apportion and the host memory access, which leads to increase the performance. In has developed a novel load balancing scheduling algorithm[6] for wireless sensor networks uses optimal scheduling algorithm for packet forwarding which determines the time slot for sending the packets for the nodes. The algorithm provides uniform packet loss probability for all the nodes. The algorithm uses balanced cost objective function for optimum scheduling. In has developed the algorithm[7] for the load is spread out evenly to all nodes. The load is divided in order of round robin; where the equal load is appointed to every node in order of circular in absence of priority and it will be send back to the initial node if the destination node has been reached. Every node keeps its allocation of index load from host node. Round robin algorithm is easy to execute, simplex and free starvation. It does not require inter procedure information and it also gives better performance for the purpose of applications. It is unable to give the assumption result and then the processing time of jobs is unequal. In

has developed an algorithm for selecting a node based on random selection, without any current or previous load on the node information. The Randomized algorithm[8] is basically a nature of static, which is suited for the equal workload on every node. Every node keeps its own record of work load, hence there is no inter procedure information is required. It may cause a congested on the single node, while the remaining node is not congested. In have introduced local queue algorithm[9] to support the inter process migration. The idea of local queue algorithm is static allotment of all process migration of new process, which is initiated by the remote when its work load fails under the number of ready processes. When the remote gets burden, the remote host receives the request for the activities. The hosts then search the data in its local list for the ready activities and the remaining activities are forwards to the requestor host and the host receives the acknowledgement. This is a distributed co-operative algorithm[10]. It requires inter procedure information but smaller as compared to algorithm of central queue. The parameters used in load balancing are given in Table 1.

Table 1. List of parameters

Parameters	Description
Nature of load balancing	It identify the algorithm behavior i.e. whether static or dynamic
Throughput	It gives the details about the completed tasks after its execution.
Process Migration	It is used to give information about time taken for the transfer of process from one node to another node.
Response Time	It is the time taken to complete the process which is for load balancing algorithm.
Performance	It is used to check capability of the system.

2. Proposed Model

Initially, client has sent their host id to the main host server for accessing the resources in the main host table. Server will accept the request from the client and it is added to the registered client host list table. All the required client ID and client address are stored in the database. Only registered client allows the mobile agent to go through the different modules. Registered client will remove any unwanted mobile agents if they try to replace the original resources. Table 2 shows the registered client host list, Mobile agent will be specified during the searching of resources. During the search of specific resources, the mobile agent and data size will be given as input and the required operations will be performed. If the required operation is performed, workload time and process migration are calculated and updated in the database. Otherwise, the client request will be dispatch to another registered client. A general architecture of the proposed model is illustrated in Figure 1. On another task type, load will be distributed among the client host depending on the amount of resources is available. Mobile agent and data size is specified in the task manager. Computational task will schedule for all the operations in the server host task manager. A required operation will be selected based on the number of priority given by the registered clients. Example: Suppose if the priority for the addition operation is higher than other operation, Addition operation will be selected first and the result will be displayed. Workload and response time is updated for the required operations. Similarly, the remaining operation on given specific task should be performed and the given data size as input. The data is distributed among the given number of mobile agents and its result should be updated in the Client request Table. The total time and elapsed time will be stored in the database.

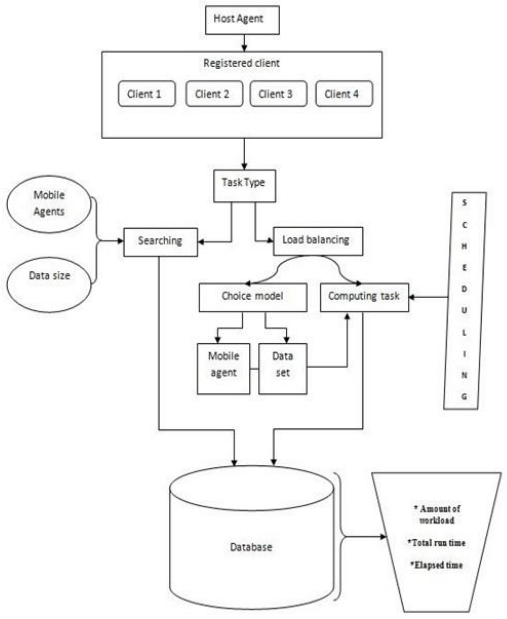

Figure 1. Architecture.

Table 2. Registered client host list

Host ID	Host Address
1	atp://localhost:2010
2	atp://localhost:2011
3	atp://localhost:2012
4	atp://localhost:2013

2.1 Resources Distribution for the Registered Client

Client host will send its host address to the main host for registration in order to provide access to the main host resources. Once the main host accepts the entire client hosts address and the accepted client host address will be maintained in the main host table (database) along with the separate workload field for each client host address (i.e.) stored in the main host table. Initially the work field will be set as 0. After certain of operations, the work load field will be updated in the main host table. The Overview of resources distribution to the clients is illustrated in the Figure 2. If any unregistered client host tries to access the main host resources, the main host denies the request using the "Aglet Proxy". In an Aglet System, it introduces the new interfaces as Aglet Proxy, which will perform as act on aglet and it provides a mutual way of obtaining the aglet. In an aglet class, which contains a several public methods that class should unable to access directly from other aglets for their reasons of security, if any aglet that needs to transmit with other aglets must first incur the proxy of aglet and then it will communicate through this interface. In other words, the aglet proxy represents as a hide object that covers an aglet from malicious aglets. When invoked, the Aglet Proxy object consults the security manager to determine whether the current execution context is permitted to perform the method.

2.2 Searching the Required Resources using its Authentication

The entire client list has been registered in the database first. After that, the searching process will be only processed among the registered clients. By specifying the number of mobile workers and the data size, the search key will be automatically generated along with given amount of workload. When the required resources is not available in the respective client host, the message will be displayed as "found" along with the total number of keys. Because of this process workload in the main host table will incremented based on the given input data size. It also includes the results of start time, end time and total run time. The overview of authenticated search is illustrated in Figure 3.

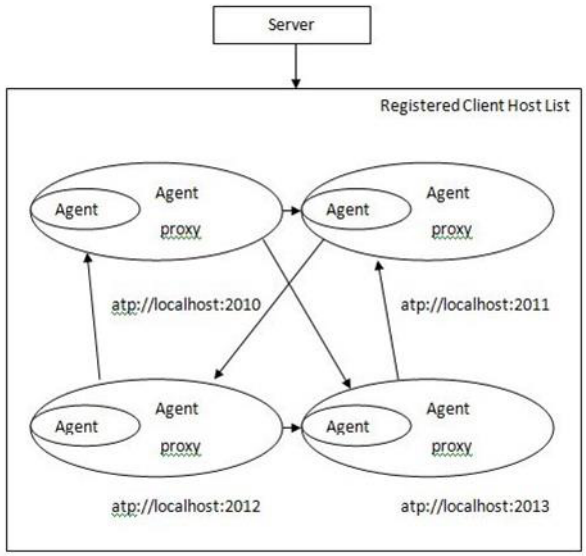

Figure 2. Overview of resources distribution to the clients.

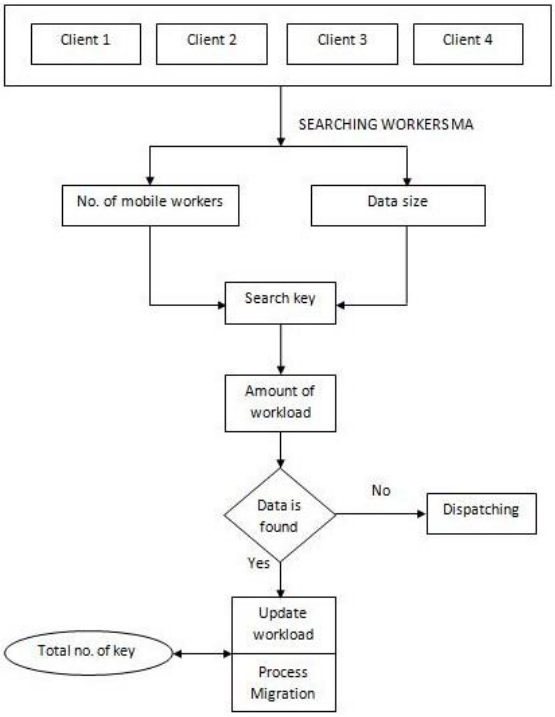

Figure 3. Overview of authenticated search.

2.3 Load Distribution by Different Choice Model

In an aglet system, in order to distribute the load effectively through mobile agents can be achieved by using efficient scheduling algorithm. By choosing this specific choice model along with the required task, the resources will be distributed equally to various registered client host. For example, when the given choice model is Loop Model, the matrix will be formed by specifying the row and columns along with number of loop count and the number of mobile agents. On specifying the required operations like addition, matrix will produces the result by taking the total number of agents and data in the matrix. In the Random Model, they have to specify the number of mobile agents (workers) along with the matrix row and column. It will generate input of A and B with the given computational task (for e.g.; addition). The events have been processed for the given Addition matrices of random model. It will display the amount of workload for each worker along with client ID. The given workload will distribute to all registered clients by using scheduling algorithm. Similarly, for the entire choice model they will specify the number of mobile agents and its related matrix row and columns. Then, we have to choose the computing task for given choice model for them scheduling the workload by using efficient scheduling algorithm. By using this algorithm, the given data size is distributed to the registered client. After the resources are fully distributed through the registered client hosts, the main host server will update the workload in the main hash table, processing time (start time, end time, total run time and total elapsed time) between the given numbers of mobile workers.

3. Experimental Results

During the searching process, message is displayed only for registered clients (URL:atp://localhost:2010, URL:atp://localhost:2011, URL:atp://localhost:2012, URL: atp://localhost:2013). In the searching technique, when the number of the agents and total data size is given as input. Then the total work load time, total amount of workload, starting and ending response time is calculated. As the number of the mobile agent's increases, total time also increases. In the proposed system, task type of searching displays as a sample output with different client_ID, total time, start and end time in Figure 4. During the task type of load distribution phase, client_ID is given starting time is calculated. Client id sent a request to the load manager for accessing the resources. The number of the mobile agents is specified, total time, elapsed time and worker run time is calculated, which is shown in Figure 5. With the help of aglet system, the performance is increased. Finally the resource is distributed equally to all the registered clients and the workload is updated along with the host_id and host_address given in the main host table Figure 6.

Client_ID	Task_Type	NoMobileAgent	Total_Time	ElapsedTime	Type_Time	Worker_Total_Time
1	AddLoopMatrixes	4	3541	5179	milliSec	689
2	AddLoopMatrixes	2	2527	4360	milliSec	143

Figure 4. Searching task type – client request table.

Client_ID	Task_Type	NoMobileAgent	Total_Time	Type_Time	Start_Time	End_Time
1	Searching	2	-13135398067	Sec.Tenths	11:1	11:1
2	Searching	4	-11452437951	Sec.Tenths	10:18	10:18
3	Searching	3	-12143872486	Sec.Tenths	16:31	16:31
4	Searching	5	-10239846278	Sec.Tenths	14:54	14:54

Figure 5. Load balancing task type – client request table.

HOST_ID	HOST_ADDRESS	Work_Load
1	atp://localhost:2010	202374
2	atp://localhost:2011	202390
3	atp://localhost:2012	300112
4	atp://localhost:2013	300112

Figure 6. Host table – work load.

4. Conclusion

Agent application can occur in an uncontrolled and heterogeneous function. Some of the existing technique shows us how to achieve load balancing through mobile agents. By using aglet system, a Load balancing is achieved by distributing the resources to all registered client host from the main server host database. If any unregistered client host tries to access the resources main server host will deny the request. After the required operation is done the corresponding workload, process migration and response time are calculated and updated successively. Hence with the help of aglets the performance is increased by enforcing load balancing without compromising on security.

5. References

1. Analysis of Large-Scale Topological Properties for Peer-To-Peer Networks. Date Accessed: 19/04/2004. Available at: http://ieeexplore.ieee.org/document/1336545/.
2. Danny B Lange. Mobile Objects and Mobile Agents: The Future of Distributed Computing. Springer Berlin Heidelberg; 1998 Jul. p. 1-12.
3. Danny B Lange, Mitsuru Oshima. Mobile Agents with Java: The Aglet API, World Wide Web. 1998 Sep; 1(3):111-21.
4. Danny B Lange, Mitsuru Oshima. Programming and Deploying JAVA Mobile Agent with Aglets. 1st Edition, Addison-Wesley Professional, 1998 Aug.
5. Daniel Grousa, Anthony T. Non-Cooperative Load Balancing in Distributed Systems, Journal of Parallel and Distributing Computing. 2005 Sep; 65(9):1022-34.
6. Laszlo E, Tornai K, Treplan G, Levendovszky J. Novel Load Balancing Scheduling Algo. for Wireless Sensor Networks, CTRQ 2011: The Fourth International Conference on Communication Theory, Reliability, and Quality of Service IARIA; 2011. p. 54-59.
7. Abubakar, Haroon Rashid, Usman. Evaluation of Load Balancing Strategie, National Conference on Emerging Technologies; 2004. p. 67-70.
8. Mohsen, Hossein Delda. Balancing Load in a Computational Grid Applying Adaptive, Intelligent Colonies of Ants. Informatica. 2008; 32:159-67.
9. Sandeep Sharma, Singh S, Meenakshi. Performance Analysis of Load Balancing Algorithms. World Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Automation, Control and Information Engineering. 2008; 2(2):367-70.
10. Mueen Uddin, Jamshed Memon, Raed Alsaqour, Asadullah Shah, Mohd Zaidi, Abdul Rozan. Mobile Agent based Multi-layer Security Framework for Cloud Data Centers, Indian Journal of Science and Technology. 2015 Jun; 8(12):1-10.