IOSR Journal of Mobile Computing & Application (IOSR-JMCA)
e- ISSN: 2394-0050, P-ISSN: 2394-0042.Volume 8, Issue 5 (Sep. – Oct. 2021), PP 09-20
www.iosrjournals.org

Effective Network Traffic Management In A Heterogeneous Servers Cluster Using Hybridized Central And Mobile Agent- Based Approaches.

Ikporo, Stephen Chibueze (Ph.D)[1], Ugah, John Otozi (Ph.D)[2]
[1] *Department of Computer Science, Ebonyi State University, Abakaliki – Nigeria*
[2] *Department of Computer Science, Ebonyi State University, Abakaliki – Nigeria*

Abstract
Network service providers nowadays consist of heterogeneous servers which makes traffic management complex and requires a high-level algorithm. For effective traffic management, there are a lot of data collection, analysis and real-time decision making required. Most existing algorithms like centralized, Round-Robin, etc, lack decision making intelligence for effective traffic management. This causes traffic congestion in the server stations resulting to service failure and denial of service, especially in a large network. The Simple Network Management Protocol for example uses the clients-servers model, where servers station act as clients that provide users' interface to the network manager and interact with agents, which manages remote access to workstations. This interaction generates significant inter-process traffic that overloads the server stations. Hybrid Mobile Agent-based approach combines centralized and mobile Agent-based approaches to provide an effective algorithm with mobile agents gathering and analyzing server status information and transmit same to the central server for traffic management decision making to reduce overload. The mobile agents move across servers within the networks to ensure even distribution of loads and also migrate loads from overload to less loaded servers thereby reducing response time and traffic congestion which improves the efficiency, rellaibility and quality of service in the network.

Keyword:—Congestion, Load -balancing, Mobile -Agent, Network- traffic, Quality –of- Service

---

Date of Submission: 20-09-2021 Date of Acceptance: 05-10-2021

I. Introduction

Network traffic management basically comprises of collecting and analyzing of data from connected devices, controlling and monitoring all devices in the network and managing the whole network from a single point. One important functional area of network management is performance management. This involves gathering statistics about network traffic and schemes to condense and present data as to balance the load on the network. Present network performance management systems such as Simple Network Management Protocol (SNMP) for data networks, and Common Management Information Protocol (CMIP) for telecommunication network, etc use centralized technique, and are characterized by lack of distribution intelligence, re-configurability, efficiency, scalability, fault tolerance and a low degree of flexibility [1]. These network management algorithms deal only with data gathering and reporting methods; which in general involves substantial transmission of management data, which causes a lot of bandwidth consumption, computational overhead; network strain and traffic jam at the manager host. These management activities are limited, and cannot take intelligent processing such as judgment, forecasting, real-time decision making, analysis of data, and positive maintenance of quality of service. Therefore, all these problems are a pointer that network traffic management requires mobile agent with intelligence to overcome the problems of network delays and information bottleneck at the centralized management station and meet today's requirements techniques.

Various applications, particularly those related to multimedia, require guaranteed transmission of data with a certain degree of reliability and quality of service in a communication network. The primary task of a network management system is to manage a network, undertake its proper functionalities, maintenance, security control, gathering and archiving of data and fault management [2]. The need to improve the performance of the distributed web servers demands the introduction of a Mobile Agent into the network which will be able to study the execution time of each load, determine the availability status and as well the state of the servers in order to migrate loads from the overloaded servers to less loaded or utilized servers. This increases efficiency, fault tolerance, high performance, and robustness, reduces network latency time and provides for asynchronous and autonomous load transmission over the network. Agents are sophisticated computer programs that act

autonomously on behalf of their users, across open and distributed environments, in order to solve complex problems [2].

Presently, mobile agents can be deployed to manage traffic in today's large heterogeneous networks. They are special software objects that can migrate from one node to another carrying logic and data, performing actions on behalf of the user. In Mobile agent-based network traffic management, agents are equipped with network management capabilities and are allowed to issue requests to managed devices (or nodes) after migrating to these nodes. Mobile agents give the flexibility of analyzing the managed node locally. Instead of querying the managed node for every fixed interval and analyzing the performance from management station, mobile agent can be dispatched to analyze the node locally [3].

II. Literature Review

Definition of a Heterogeneous Network: According to [4] and [5], defined Heterogeneous network as a distributed system that consists of set of various servers with the same functionality but different processing capabilities. The fundamental problem associated with this in terms of load balancing is the choice of static and dynamic load balancing policies. The nature of heterogeneous network is that the servers involved have different speed, communication link speeds, memory sizes, and variable external loads due to the variation of the servers. Every server in the network acts as a server having same copies of data and resources that may be requested by the users.

Heterogeneous versus Homogeneous servers: The heterogeneity or homogeneity in server could be software based or hardware (machine) based. Whether the heterogeneity of the server is software or hardware, their performances and output always vary [6] and [7].

Heterogeneity Server based on Software: According to [6] and [7], the software-based heterogeneity of server can be seen in terms of the platform with which they are running on. In this case, the variation in the operational platforms are considered and seen as major factor that affect the performance and the output of server. These operational platforms include Internet Information Services (IIS) and Apache platforms.

Apache Server Platform: Apache always known as the Apache HTTP web server is an open source Web server application that is managed by the Apache Software Foundation. It is freely distributed, and its open source license means users can edit the underlying code to tweak performance and contribute to the future development of the program [8]. Although Apache will run on all major operating systems, it is mostly used in combination with Linux. The two, combined with MySQL database and PHP scripting language, comprise the popular LAMP Web server solution [9].

In terms of performance, Apache is better than IIS. However, in most cases, Apache is still held back by two of its main features:

a. Feature bloat: Apache is seen to be an extremely feature-rich application with users only using about 10% of the features on a regular basis.

b. Apache is a process-based server: In a process-based server, each simultaneous connection requires a separate thread and this incurs significant overhead. An asynchronous server, on the other hand, is event-driven and handles requests in a single or very few threads [10]

Internet Information Services (IIS) Server Platform: Internet Information Services platform is Microsoft's web server offering and are second to Apache. It is a core Microsoft product and is bundled and runs on Windows operating systems. It is a closed software product and supported solely by Microsoft. It has a newer version called IIS Express that has been installable as a standalone freeware server from Windows XP SP3 onwards. But this version only supports http and https [11] and [12]. In terms of feature, performance and security, there are solid improvements with IIS when compared to Apache. Security especially has been one area of significant gain, making huge leaps from the days of IIS 6.0's vulnerability to the infamous Code Red worm. Although, IIS has been called out as still being poor at supporting PFS (Perfect Forward Secrecy) – a property of key cryptography that ensures a long-term key, it will not be compromised if a single component session key is compromised or broken. IIS vulnerability is also largely blamed on its operating system parent since most malware targets Windows, and Linux (Apache's main choice of Operating System) is itself an offshoot of the inherently iron-clad Unix OS [12].

Heterogeneity Server Based on Hardware (Machine): Heterogeneity of server could be based on the host machines. In this case, server machines are heterogeneous if there is variation in Central Processing Unit processing speed, Hard Disk type and transfer rate, bandwidth usage and RAM. This components variation plays a big role on the rate of job processing and overall performance of the servers and their output. However, when the server makeup is the same, they are called homogeneous servers. In this case, the variation of their job processing rate is always the factor of job requirements and server reliability [13] and [14].

Importance of Network Traffic Management in Heterogeneous server: Network Traffic management is very essential in distributed computing system with heterogeneous servers in order to improve the quality of service by managing customer loads that are changing over time. The request demands of incoming requests are optimally distributed among available system resources to avoid resource bottlenecks and to fully utilize available resources [15].

According to [16] and [17], Network Traffic management heterogeneous server network is an active technology that allows for shaping, transforming and filtering of the network traffic by routing and balancing them to the optimal server. Traditional load balancing methods are implemented in an open network based on message passing paradigm [18]. Current network management systems such as SNMP and CMIP, etc. have flaws as they are only limited to distribution of workload to various servers and ensuring that they get to their respective destinations using the specified technology inbuilt in them with no consideration to the processing capabilities of such servers and as well the execution time required by the distributed loads, which determines the availability or otherwise of connected servers. This does not allow for scalability, availability, fault tolerability and quick response time especially as the servers are with varied in their operational capability. Hence, the simple server granting these services as provided by these approaches make the traffic in the network vulnerable to failure at the peak traffic time.

The importance of traffic management in a heterogeneous server network is that it helps to increase availability, reliability, optimize resource utilization, improve throughput, maintain stability, and provide fault tolerant capability, minimize response time and avoid overload of any simple resources. This improves the quality of service by managing customers' loads and ensuring that all the incoming requests are optimally distributed among available system resources to avoid resource bottlenecks and as well ensure full utilization of available resources [15].

In a heterogeneous server farm, network traffic management ensures workload distribution across multiple computing resources within the network cluster, such as computers, computer clusters, network links, central processing units, or even disk drives. It usually involves dedicated software or hardware, such as multilayer switch or a Domain Name System server process. Load balancing divides traffic between network interfaces on a network node (OSI model layer 1) basis [19].

With the mobility and autonomous properties of mobile agent, there exists enormous amount of potential areas in which they may be of great assistance. Mobile agents are never restricted to local execution environments, nor forced to communicate through structured but robust mechanisms rather, they have the ability to halt execution, travel to another host and resume its execution there. Hence, the network is at its disposal. This therefore, represents a great leap forward in terms of distributed systems and information management. Their ability to migrate computation toward resources, and agents which notifying the user of the changes in the network environment and dispatching the agent to carry out some tasks offline, makes them not only outstanding and unique, but also more advantageous compared to other software. They can transverse the Internet, often on behalf of their user. [20], In their work postulated that intelligent search agents access dynamic information in heterogeneous networks. Mobile agents are considered to be effective in many applications such as reduction of network workload and network latency. They can help reduce the network bandwidth consumption by creating agents which handle transactions, and sending them to where data resides. This ensures that instead of intermediate results and information passing over the wire, only the agents need to be sent. This will definitely help to solve the problem in the client/server network bandwidth which is valuable resources in distributed applications, [21]. Therefore, the slogan of mobile agent application is: move the computations to the data rather than the data to the computations, [22].

According to [21], mobile agent architectures can solve the problems caused by intermittent or unreliable network connections. With their asynchronous and autonomous execution properties, mobile agents can be dispatched into the network where they work independently and return results when the host reconnects to the network, [23].

Mobile agents are capable of sensing their environments and reacting dynamically to changes which can make them useful especially in intrusion detection. They can move around to gain better position or avoid danger, clone themselves for redundancy and parallelism, or marshal other agents for assistance. Hence, mobile agents can assist to build robust and fault-tolerant systems, [3].

Mobile agents reduce design risks, and the decision about the location of code (client versus servers) can be pushed toward the end of the development and the system can be easily modified after it is built and in operation, [21].

They are generally independent of computer system and transport layer despite network computing been fundamentally heterogeneous. Hence, they have the potential to provide optimal conditions for seamless system integration, [23].

According to [24], mobile agents provide the following advantages:

- **They save spaces**: Mobile agent code and state do not need permanent storage in the hosts they run.
- **They are asynchronous and autonomously Interactive**: Mobile agents can be delegated to perform a certain task and the agent is intelligent enough to decide at execution time, with whom it needs to communicate.
- **They reduce network traffic**: Mobile agents are based on the concept of bringing computation to data rather than data to computation.
- **They provide robustness and fault tolerance**: Mobile agents can be used to increase availability of certain service by assigning individual agents for each service.
- **They support Heterogeneous Environment**: The mobility framework support of mobile agents allows them to be separated from the host and its operating systems.

Centralized/Manager Server-Based Newtork Traffic Managemnet Approach in a Cluster Network: According to [25], centralized server approach allows for the provision of a centralized server for every cluster which is not assigned with any workload at the beginning of the process. This central server serves as a dummy server where loads are assigned each time the overloaded server is not be able to find the lightly loaded server in the cluster to transfer loads. Here, whenever a particular server is overloaded, the manager server first searches for other lightly loaded server within the cluster; and when found, the load is then transferred to the found server, thereby balancing the system loads. However, if no other lightly loaded primary server then the centralized server takes notice of wait for free server to transfer the load. The centralized server is made to have some better configuration structure than other server in the cluster. The network traffic between centralized server and all other servers are kept at minimal level in order to avoid network delays. With this arrangement, any overloaded server can easily reach the centralized server and transfer the load easily in case it is heavily loaded. According to [26], the central server is responsible for making the load balancing decision and with the information required for the process being obtained from the remaining slave servers, usually on demand basis or within a predefined fixed time interval. The need for the system load information may also be gathered only when the system's current load status changes. According to [27], the central server always select the server with least load when compared with other servers in the cluster whenever new request is created and need to be assigned. The central server maintains the load status of all the servers within the cluster which it stores locally. Whenever a server's load staus is changed, a message is sent by such server to the central server to keep it updated. However, the need for all the servers within the cluster to update the central manager with their load status leads to more inter process communication which may cause network congestion and it has scalability limitation as the major setback, [28].

A particular server can become unavailable due to failure or planned/unplanned downtime. At this situation, it is the duty of the central server to take over immediately from the failed server. The central server can as well function as a failover server and performs no other function other than taking over from failed servers, [29]. This ensures that at every point in time, workloads are distributed to the computing devices to avoid long wait by the central server even without the knowledge of the client/user. This ensures that the clients don't experience any downtime during the operation, [30].

Mobile Agent and Network Traffic Management: The introduction of Mobile Agent (MA) into network traffic management in a heterogeneous cluster server environment provides new technology with which an autonomous program can migrate on its own or host controlled from one server to another in a network. This means that a program running in a host can suspend its execution at any point and transfer itself to another host (or even request the host to transfer itself to its destination) and then resume execution from the same point of suspension, [31].

Mobile agents are designed to collect the load information of the various heterogeneous servers proactively and communicate the load information to the central server. The mobile management unit in the central server receives the load information and computes the rank of the server. The least loaded server is given a rank of one. Based on the rank of the server, the mobile management unit updates the dispatcher table in the dispatcher. The dispatcher table is dynamically updated and it is used by the dispatcher to migrate load to the server using the specified policy thereby balancing workload in the network, [32].

Mobile Agents are autonomous programs that are proactive, reactive, and can migrate freely from one host to another in heterogeneous networks, transporting its state and code from home host to another and executing various operations on the network, [33]. They interact with each other as they move from one server to another sharing information and spreading intelligence across the network, [34].

According to [15], Load balancing is very essential in distributed computing systems in order to improve the quality of service and the management of client requests which continue to change. It allows for all incoming requests optimally to be distributed among the available servers to avoid resource bottlenecks as well as to fully utilize the available resources. Mobile agents help in managing the network by ensuring that the loads on the network are distributed evenly to all the servers so as to avoid server being idle or overloaded. This mobile agent-based model provides high flexibility, efficiency, and low network traffic, less communication latency time and will be highly asynchronous. They are designed to undertake intelligent processing such as judgment, forecasting, decision making, data analysis and maintenance of quality of service which the management agents do not provide, [35]. They can suspend their execution at any arbitrary point and jump to another server and resume execution there. Their mobility features allow them to be created, deployed, and terminated without disrupting the network configuration, [1].

According to [36], Mobile agents could be described as autonomous intelligent software entity capable of reasoning, migrating and running in another remote node within a heterogeneous network, which is capable of searching and gathering results in cooperation with other mobile agents, and returning to the home site after completing the attributed tasks. It performs the user's tasks by migrating and executing on server hosts connected to the network. Each agent is typically composed of the agent code, the agent execution thread along with execution stack, and the agent data part, which correspond to the values of the agent's global variables. Mobile Agents differ from remote procedure call approaches because, in mobile agent system, the mobile agents moves to where the data source resides and resumes execution there while in remote procedure call systems, the data is brought to the running process after a remote procedure is executed. Again, with mobile agent systems, agents move and execute tasks in remote hosts when they choose whereas in the other approaches, the process is executed by other programs that may reside on the side of the communication link. Hence, mobile agent approaches is much faster than a remote procedure calls. They additionally possesses basic agent model and navigation model including life-cycle model, computational model, security model, and communication model with which software agents are recognized, [24].

Mobile Agents and Network Load Balancing in a Heterogeneous Server Network: Mobile agents in heterogeneous server network control network's devices and save the manager capacity and network bandwidth. Since mobile agents are programs being sent across the network from client to the server or vice versa, they can travel in network following their itinerary and carrying logic and data to perform a set of management tasks at each of the visited nodes in order to meet their designed objectives, [37].

According to [38], Mobile agent technology provides solution to the various load distribution problems in a heterogeneous server with different computational abilities, where process migrations are traditionally under the supervision of centralized controller. With this technology, multi agents are responsible for the decentralization of computational load distribution. A complex application can be divided into autonomous parts, each of which delegated to one or more mobile agents. Each of the mobile agents is in charge of searching for the node of the network which offers the most convenient resources, where to execute its own part of code. These agents can move to other node where more computational resources are available for more efficient load distribution. These mobile agents are capable of cooperating with other agents in the network in order to perform complex or dynamic tasks. They can read from and write to a shared blocked of memory on each node, and can use this facility both to coordinate with other agents executing on that node and to leave information behind for subsequent visitors.

The mobile agent-based load balancing technique ensures that no server is idle or overloaded and allows network use all its capacity efficiently. It provides ways for managing the network resources in a well-organized approach. It allows for better use of the exciting network infrastructure, improves service to the end users and reduces the cost of providing these services. The idea is to provide quality of service with dedicated bandwidth, control jitter and avoid latency, which are required by most real time applications, and for recovering loss qualities, [39]. They allow the transformation of current networks into remotely programmable platforms and they are solutions for managing distributed networks.

According to [40], the concept of mobile agent-based load balancing is a better alternative to the traditional client-server programming based on the remote procedure call or the static distributed object paradigm. The use of mobile agents in the management of telecommunication network helps to reduce network traffic by distributing workloads and building scalable and reliable distributed network management system. And to perform optimally, they must communicate to find their peers, to cooperate and negotiate in open environments.

Mobile agent-based load balancing allows for the transfer of the administration tasks to the mobile agents. With this, the network management tasks and computational load are distributed instead of being centralized towards and on the manager hosts, which will ensure quick network connections without noise and without several trials. This will also ensure that the network will be organized in order to work professionally to

adjust to changes successfully and react to problems such as traffic patterns. The use of mobile agent in network load balancing also enables decentralization and potentially solves most of the problems that exist in centralized server/client solutions. This therefore makes such applications to be more scalable, more robust, and can easily be upgraded or customized, [40].

Mobile agent-based technology helps to design a network architecture that is flexible. This is because; they are distributed and can scale upwards in size quite gracefully. Also, because agent populations can change over time, new usage contexts and models can be accommodated. And with the system interaction mediated by the agents, multiple network management strategies can coexist and co-evolve. This helps system engineers to create building blocks for networks which can embrace changes in network size, congestion and proffer user needs, [41].

Mobile agent based helps for quick update of the routing tables and allows for easily modification of the selection criteria. This is done by emitting the corresponding agents which will perform some required modifications in the routing table while considering a given subset of the criteria. This increases flexibility in addition to the modularity of the architecture because of the transparency in the modification of the load agent and strategy agents.

III. Discussion

Components of Mobile Agents System: Mobile Agent Systems depending on what they are set to achieve are usually composed of some components which helps it function very well. Basically, there are three components of mobile agents-based system. They include:

i. Load Information Agent.
ii. Location Information/Job Delivery Agent.
iii. Routing Information/Server Management Agent.

i. Load Information Agents (LIA): This works like the information policy in dynamic load balancing algorithm. They are responsible for monitoring the state of each node in the network. They migrate to the shortest next destination node when the node is slightly or moderately loaded. In the same way, if the state of the node is heavily loaded, it (switches to the next mode) launches location agent in all nodes with x – hops far from the current node. If the routing information of nodes with x hops far from the current agent is not available the information agent then launches location agent in all nodes with x – 1 hop far of the current node. And again, if it is not found, it continues decreasing the number of hops until, it arrives to one hop.

ii. Location/ Job Delivery Agents (L/JDA): This plays the role of the location policy component of any dynamic load balancing algorithm. The role of the location agent is to find suitable receiver partner for the overload-node that launched it.

iii. **Routing Information/Server Management Agents (RI/SMA):** This is responsible for updating the routing table which resides at each node. It carries a routes vector table containing the communication cost from the assigned node to each other node in the cluster. This table has a lifetime measured by the number of the hops the routing agent is allowed to perform before updating the vector table. The routing agent plays an important role in informing each node in the cluster about the addresses of other nodes and if a failure of a node or a link is detected, it is the role of the routing agent to spread it over the cluster by copying the new table and migrating. However, it is important to consider the propagation delay to move to each node in the cluster before the new table arrives to all other nodes.

Architecture for Mobile Agent Design Platform in Traffic Management: Mobile Agents are normally designed using a special framework called Platform for Mobile Agent Development and Execution (PMADE). The design architecture is such that it can consist two component parts, Agent Host (AH), which is responsible for accepting and executing incoming request of autonomous java code and an Agent Submitter (AS), which submit to the mobile agent on behalf of the user to the AH. The AH is the key component of PMADE architecture and consists of the manager modules and the Host Driver as shown in Figure 1. The Host Driver lies at the base of the PMADE architecture and with the manager modules residing just on top of it. It is the basic utility module responsible for driving the AH by ensuring proper co-ordination between various managers and making them work in agreement.

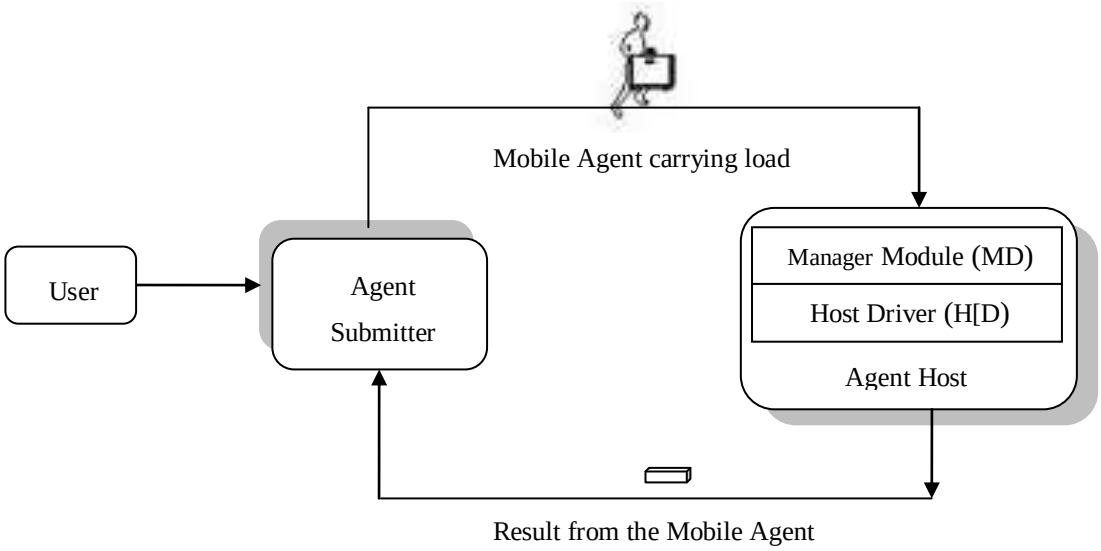

Figure 1: The Architecture of PMADE

Whenever any user wants to process a task, the task is submitted to the mobile agent designed to perform such task through the Agent Submitter on the user system. The AS then tries to establish the connection with the specified AH where the user already holds an account. If the connection is established, then AS then submits the MA to the user and then continue with that line execution. The agent host observes the behaviour of the received agent then executes it. The agent execution depends on its environment and its state within which the transformation of agent can be made from one AH to another whenever it is required. Whenever the execution is completed, the agent then submits the result to the AH, which then stores the result until the remote AS retrieve them for the user.

Operational Mechanism and Implementation Architecture of Mobile Agent: The operational mechanism of mobile agent is usually designed on the framework called Platform for Load Balancing (PLB). The architecture consists of three components: the Agents, Load Balancing Policy and the Data Bank as shown in Figure 2.

The traffic management policies are the guiding rules that guide the mobile agents in taking load migration decision. The data bank is the data storage for all the entries made through the input interfaces. The mobile agent make use of content based technology and knowledge based engine that is trained and retrained to study the processing speed and availability status of each server. The Mobile Agent was designed using the PLB framework. The implementation was based on the Platform for Mobile Agent Distribution and Execution (PMADE).

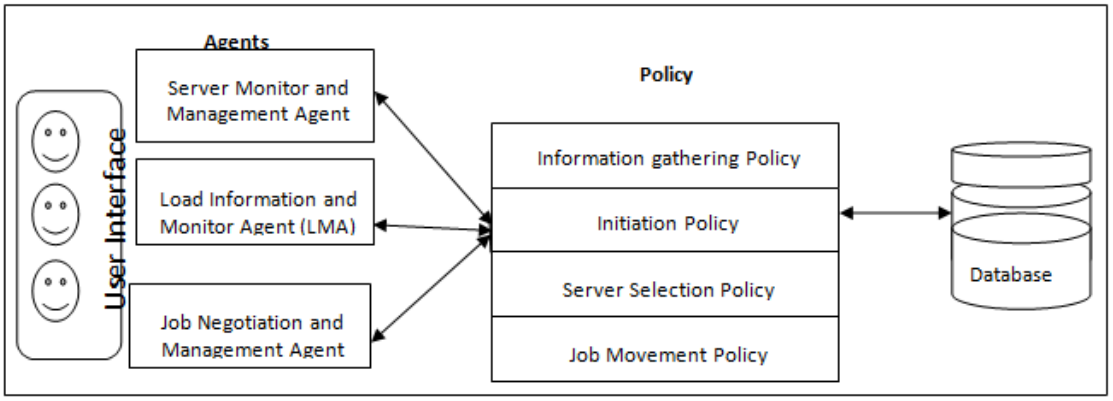

Figure 2: Architecture of PLB

Enabling features of Mobile Agent for Network Load Balancing: Mobile agent paradigm has emerged the exciting paradigm for mobile computing applications. Mobile agent technology helps in the design of wide range of adaptive, flexible application with non permanent connections by adding mobility to code, machine-based

intelligence, improved network and database possibilities, [42]. The properties which make mobile agents suitable for design of Mobile agent-based network load balancing include the following:

i. Configuration Management: Mobile agents employ their mobility characteristics and dynamic adaptability in software configuration management. In network cluster, many servers are normally connected together and each time there is need to deploy dynamic software which requires reconfiguration with the new software version or data set. [43], stated that mobile agents could be used to convey the new software to all the connected servers. A scalable runtime environment is provided for mobile agents to enable them fit into various types of network.

ii. Traffic Management: Mobile agents can proactively and autonomously carry out administrative task such as installation and upgrading of software and periodic monitor of the network. They can be deployed to effectively decentralize network management functions, [44]. They can be dispatched to the remote host containing data and performing computation there. This allows for computations to be moved to the data storage locations instead of moving data to the computing location. This reduces network traffic congestions and improves quality of service.

iii. Intelligence: Mobile agents employ techniques from field of artificial intelligence which enables them with some degree of intelligence and common-sense. This also makes them to be smart at arriving at decisions without compromising user's preferences. This reduces response time in service delivery.

iv. Autonomy: Mobile agents because of their autonomous feature and nature can decide on the sequence of actions to be performed to achieve the user's target. This feature enables them to operate without human intervention. They only require the specification and they will proceed without the user further monitoring and commanding them.

v. Robustness and Fault tolerant: Mobile agent can act and react dynamically in presence of unfavourable situation thereby making it easy to build a robust and fault tolerant distributed and cluster network with them.

vi. Responsiveness: Mobile agents perceive their immediate environment (this can be a cluster network, collection of other agents, telecommunication network etc) and respond in timely fashion to changes that may occur in it. In the same time, they can respond to their environment and as well be able to exhibit opportunistic, goal-oriented behavior and take initiative when necessary.

vii. Adaptive Learning: Agents learn users' behavior and adapt themselves to suit the users.

viii. Communicative Ability: Agents are social entities and often communicate and collaborate with one another in order to complete their tasks. Hence, it provides a user-friendly interface so that users can easily interact with the agent.

ix. Seamless System Integration: Networking computing is heterogeneous in nature both in terms of software and hardware. According to [42], Mobile agents can provide seamless system integration because of dependence only on the environments in which they execute.

x. Persistence: The mobility feature of agents also gives them persistence attribute. As agents move from one place to another, it preserves its internal state and data. Particularly, it retains any computing results obtained during execution in the execution environment it is transported from.

IV. The Architecture of Proposed System

The proposed system combines the Centralized Server and Mobile Agent based load balancing techniques. While the central server which serves as the manager server controls and coordinates the load distribution across the network, the mobile agents undertakes the server status information gathering, load information gathering and load transfer across the server cluster in the network. The proposed system architecture consists of n number of clients connected with cloud service providers via the Internet Service Provider (ISP) and consisting of one manager/central server and n pool of shared cluster servers. At the pool of cluster servers, there are three agents; Server Monitoring and Management Agents (SMMAs), Load Information and Monitoring Agents (LIMAs) and Load Negotiation and Transfer Agents (LNTAs) see figure 3.

The SMMAs reside in each of the server in the cluster and gather server information status by constantly checking the I/O networks, memory and CPU utilization. This information is stored in the database from where the manager server gets informed of the status of the entire server within the cluster. This agent usually calculates server's availability or status using three parameters; the average CPU utilization, Memory utilization, and Network Bandwidth utilization represented respectively as CPUμ, MEMμ, and NETμ. The Server Monitor and Management Agent in each server collects the CPUμ, MEMμ, and NETμ of all the individual servers in the cluster with different tasks and stores them in the table called KnowledgeBase_Table. This information is used thereafter to categorize the server load status as: under loaded, normal and overloaded. Each SMMA uses its KnowledgeBase_table, see Table 1, to monitor its load periodically and determines the load status of the host server. SMMA in the manager server gathers the status information of every server through the Load Information and Monitor Agent (LIMA) which travel around cluster servers to get information about every

server. It is used for maintaining record of specifications of all servers in cluster. The KnowledgeBase_table contains Servers' Id, Status of its memory utilization along with CPU utilization, fitness value and load status of all servers.

Server Load Time can be calculated as follows:

Load Time (%) = CPUμ + MEMμ + NETμ × 100 - - - Equation I

Where CPUμ , MEMμ , NETμ are average utilization of server CPU, Memory, and Network Bandwidth, respectively during each given period. This is as collected by the SMMA.

Table 1: KnowledgeBase_table of the Proposed System

Server ID	**RAM**	**CPU**	**BW**	**Load Status**
Server 1	μ_1	λ_1	γ_1	Status Server 1
Server 2	μ_2	λ_2	γ_2	Status Server 2
Server n	μ_n	λ_n	γ_n	Status Server n

The server load status is gotten by calculating the estimated processing time required by each load as follows;

If Load Time (%) < 50%, Load Time = Low
If Load Time (%) = 50%, Load Time = Medium
If Load Time (%) > 50%, Load Time = High - - - Equation II

The LIMAs gather feedback from the SMMAs of each server in the database and send it to the manager server for it to know the current load status of the servers, whether under-loaded or over-loaded. This helps it to decide from which server to make a load transfer initiation and to which server. Load transfer initiation can only be made whenever there is record of overload on any of the servers. According to [43], the current load status of servers are usually catigorized into three levels: Under loaded, Medium and Overloaded. And two Threshold parameters; t-under and t-upper are used to describe the levels as follows:

Under loaded: Load < t – Under;
Medium loaded: t – Under ≤ Load ≤ t – Upper;
Overloaded : Load > t – Upper.

LNTA which reside in the manager server is made to distribute loads on the queue or transfer loads from overloaded to under-loaded servers as case may be. Once any of the servers is overloaded, Load Negotiation and Transfer Agents will be activated by the SMMA on the overloaded server.

The LNTA then executes the server selection policy to select another server and negotiates with it for the acceptance and to receive the reallocated load. This allows the sending and receiving servers to establish communication (handshake) and agree before commencing load migration. And thereafter, the LNTA transfer the reallocated load to the specified server. The LNTA can also perform load redirection on the fly. If the manager server did not establish that there is an overloaded server, load transfer will not be initiated.

The users' requests from the remote nodes are usually gathered by the manager server and stored in a queue. The manager server then checks the database to know the server status and then distribute the workloads accordingly. This ensures that workloads on the network are properly balanced to avoid server overload which leads to delayed responses and denial of service as shown in figure 3.

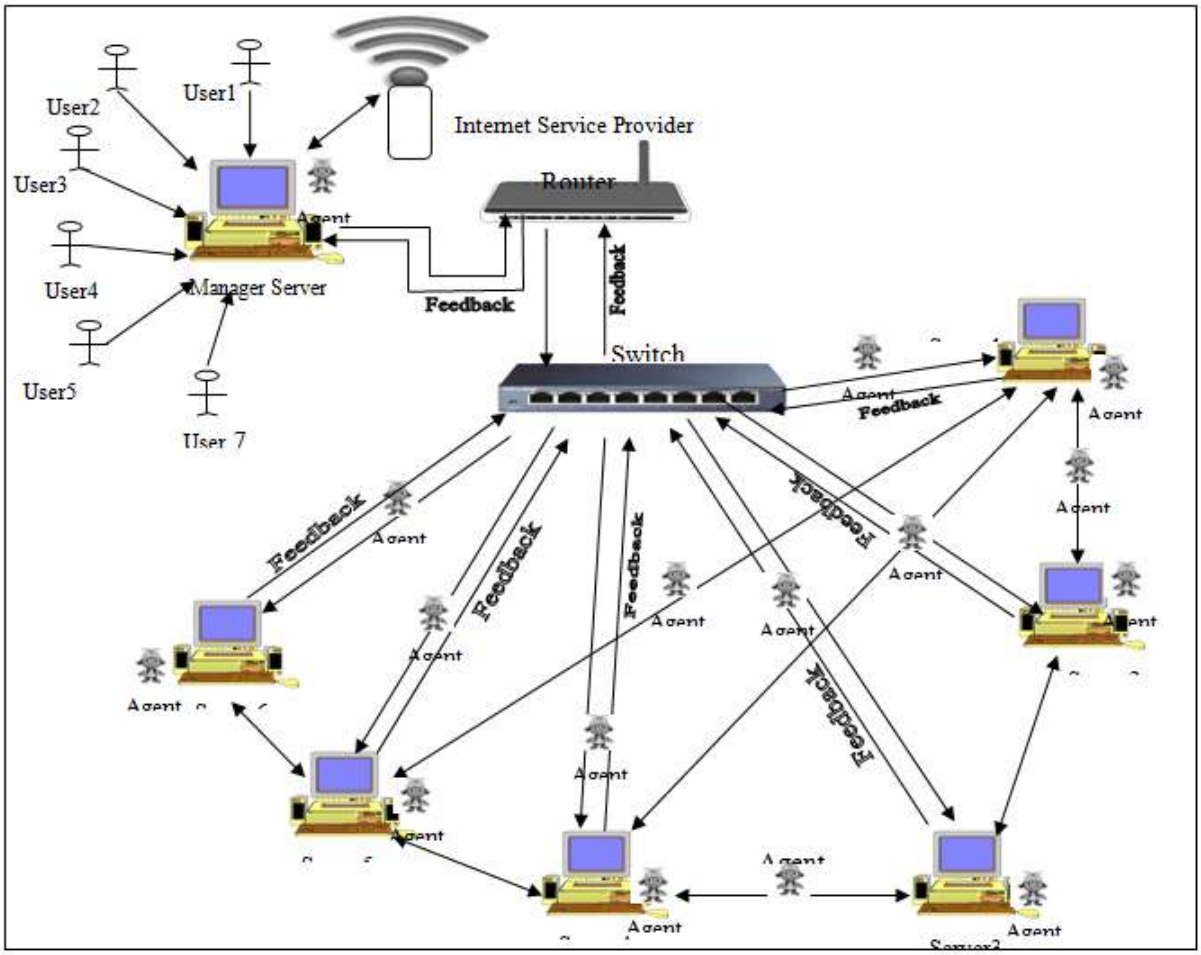

Figure 3: Architecture of the Proposed System

V. Conclusion

Network traffic management in a heterogeneous server cluster is usually complex because of the varied computational capabilities of the connecting servers. Employing the centralized server algorithm in achieving effective load balancing has over time shown some flaws such as inter process communication overhead which lead to traffic congestion and delay in service delivery. To bridge this gap, Mobile agents were introduced into the system to form a hybridized central and mobile agent-based system. With this, the manager server performs the function of load allocation and load transfer initiation; other connection servers are saddled with the duty of processing the users' requests.

The mobile agents help in checking the status of all the connecting servers, updating the manager server database and transferring of workload from overloaded servers to less loaded servers. This greatly improves the system throughput and communication overhead as the connecting servers need not communication with each other again. The real time update of the system database or dispatch table on the connecting status makes the system dynamic and intelligent. This helps the manager server to take real time decision on load allocation and also gives room for failover server whenever there is server failure thereby increasing the scalability of the system, quality of service and reduces denial of service. There is also the introduction of fault tolerance system to keep track of all the mobile agents in case any loss of and the agents on transit in order to restore or create a new one.

References

[1]. Gbanbari, M., Gavalas, D., Greenwood, D. and O'Mahony, M. (2000). Advanced Network Monitoring Applications based on Mobile/Intelligent Agent Technology. *Computer Communication Journal* 23(8), 720-730.

[2]. Ritika J. and Harjot K. (2014). Load Balancing Techniques using Mobile Agents. *IPASJ International Journal of Computer Science*, 2(12), 34-38.

[3]. Zhao, Q., Wang, H. and Zhang, Y. (2002). Comparison Study of Three Mobile Agent Systems. A Term Paper in University of Connecticut, 121-124.

[4]. Ajay, A. J., Shriwastava, S. K. (2012). Design of an Optimized Virtual Server for Efficient management of Cloud Load in Multiple Cloud Environments. *International Journal of Application and Innovation in Engineering & Management.* 1(3), 8.

[5]. Ajay, A. J. and Renu, J. (2016). Dynamic Load Management in Cloud Computing Environment. *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering.* 4(9), 186-190.

[6]. Schroeder, T., Goddard, S. and Ramamurthy, B. (2000). Scalable Web Server Clustering Technologies. *IEEE Network*, 14(3), 38-45.

[7]. Lockwood, J. W., McKeown, N., Watson, G., Gibb, G., Hartke, P., Naous, J. and Luo, J. (2007). NetFPGA--an open platform for gigabit-rate network switching and routing. In 2007 IEEE International Conference on Microelectronic Systems Education, IEEE, 160-161.

[8]. Fogel, K. (2005). Producing open source software: How to run a successful free software project. " O'Reilly Media, Inc.", 13-15.

[9]. Rosebrock, E. and Filson, E. (2006). Setting up LAMP: getting Linux, Apache, MySQL, and PHP working together, John Wiley & Sons, 55-58.

[10]. Kumar, R., Jain, K., Maharwal, H., Jain, N. and Dadhich, A. (2014). Apache Cloudstack: Open source infrastructure as a service cloud computing platform. Proceedings of the *International Journal of Advancement in Engineering Technology, Management and Applied Science*, 111-116.

[11]. Lerner, J. and Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics*, 50(2), 197-234.

[12]. Campbell-Kelly, M. and Garcia-Swartz, D. D. (2007). From products to services: The software industry in the internet era. Business History Review, 81(4), 735-764.

[13]. Forman, G. H. and Zahorjan, J. (1994). The challenges of mobile computing. *Computer*, *27*(4), 38-47.

[14]. http://www.diffen.com/difference/Apache_vs_IIS

[15]. Branko, R. and Mario, Z. (2011). Analysis of Issues with Load Balancing Algorithms in Hosted Cloud Environments. In 2011 Proceedings of the 34th International Convention MIPRO, IEEE, 416-420.

[16]. Dias, D., Kish, W., Mukherjee, R. and Tewari, R., (1996). A Scalable and Highly Available Web-Server, in Proc. 4th International Computer Conference, IEEE Computer Society, San Jose, CA, 85-92.

[17]. Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C. E., Azodolmolky, S. and Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *arXiv preprint arXiv:*14(6), 440.

[18]. Cardellini, V., Colajanni, M. and Yu, P. S. (1999). Dynamic load balancing on web-server systems. *IEEE Internet computing*, 3(3), 28-39.

[19]. Case, J. D. (1991). Management of switched multimegabit data service (SMDS) using the simple network management protocol (SNMP). In [1991] Proceedings 16th Conference on Local Computer Networks, IEEE, 362-368.

[20]. Boonk, M., DeGroot, D., Brazien, F. and Oskamp, A. (2005). Issues in Mobile in a Mobile Agent-based Multimedia Retrieval Scenario, Proceedings of the 4th International Workshop on the Law and Electronic Agents, (LEA '05). Online at: lea-online.net, 35-37.

[21]. Sundsted, T. (1998). Agents on the move, Bolster your client apps by adding agent mobility. *URL: http://www. javaworld. com/javaworld/jw-07-1998/jw-07-howto. Html*, 123-125.

[22]. Lange, D. B. and Oshima, M. (1998). Mobile agents with Java: the Aglet API. World Wide Web, 1(3), 111-121.

[23]. Lange, D. and Oshima, M. (1999). Seven good reasons for Mobile Agents. *Communication of the ACM*, 42(3): 88-89.

[24]. Bieszczad, A., Pagurek, B. and White, T. (1998). Mobile agents for network management. IEEE Communications Surveys, Fourth Quarter, 1(1), 7.

[25]. Parveen J. and Daya, G. (2009). An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Servers by Exploiting the Interrupt Service. *International Journal of Recent Trends in Engineering*, 1(1), 232.

[26]. William, L., Karypis, G., Kumar, V. and Biswas, R. (2000). Load balancing across near-homogeneous multi-resource servers. In Proceedings 9th Heterogeneous Computing Workshop (HCW 2000), IEEE, 60-71.

[27]. Andrews, G. R., Dobkin, D. P. and Downey, P. J. (1982). Distributed allocation with pools of servers. In Proceedings of the first ACM SIGACT-SIGOPS symposium on Principles of distributed computing, ACM, 73-83.

[28]. Sandeep S., Sarabjit, S. and Meenakshi S. (2008). Performance Analysis of Load Balancing Algorithms, Academy of Science, Engineering and Technology, 38, 269-272.

[29]. Chen, Y., Wen, H., Wu, J., Song, H., Xu, A., Jiang, Y., ... & Wang, Z. (2019). Clustering based physical-layer authentication in edge computing systems with asymmetric resources. *Sensors*, *19*(8), 1926.

[30]. Abdallah, M., Griwodz, C., Chen, K. T., Simon, G., Wang, P. C., and Hsu, C. H. (2018). Delay-sensitive video computing in the cloud: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *14*(3), 54.

[31]. Patel, R. B. and Garg, K. (2004). A New Paradigm for Mobile Agent Computing, WSEAS Transaction on Computers, 1(3), 57-64.

[32]. Vijayakumar G. D. and Rhymend U.V. (2014). Distribution of load using Mobile Agent in Distributed Web Servers. *American Journal of Applied Sciences*, 11(5), 811-817.

[33]. Cao, J., Sun, Y., Wang, X. and Das, S. K., (2003). Scalable Balancing in Distributed Web Servers Using Mobile Agents. *Journal of Parallel and Distributed Computing*, 63(10), 996 - 1005.

[34]. Berende, C. A. S., Ruurda, J. P., Hazenberg, C. E. V. B., Olsman, J. G., and Van Geffen, H. J. A. A. (2007). Inguinal hernia treatment with the Prolene Hernia System in a Dutch regional training hospital. *Hernia*, 11(4), 303.

[35]. Patel, R. B., Nehra, N. and Bhat, V. K. (2007). A Framework for Distributed Dynamic Load Balancing in Heterogeneous Cluster. *Journal of Computer Science*, 3(1), 14 - 24.

[36]. Outtagarts, A. (2009). Mobile agent-based applications: A Survey. *International Journal of Computer Science and Network Security*, 9(11), 331-339.

[37]. Satoh, I. (2003). Building reusable mobile agents for network management. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 33(3), 350-357.

[38]. Mohamed, B., Abdellatif, S. and Ilias, C. (2013). Load Balancing Management by efficient Controlling Mobiles Agents. *International Journal of Soft Computing and Engineering*, 2(6), 24-27.

[39]. Tripathi, A., Ahmed, T., Pathak, S., Carney, M. and Dokas, P. (2002). Paradigms for mobile agent based active monitoring of Network Systems. In NOMS 2002. IEEE/IFIP Network Operations and Management Symposium.'Management Solutions for the New Communications World', IEEE, 65-78.

[40]. Bohoris, C., Pavlou, G. and Cruickshank, H. (2000). Using mobile agents for network performance management. In NOMS 2000. 2000 IEEE/IFIP Network Operations and Management Symposium. "The Networked Planet: Management Beyond 2000", IEEE, 637-652.

[41]. Poor, R., (1997). Hyphos – A Self –organizing, Wireless Network. Master's thesis, MIT Media Lab. http://ttt.media.mit.edu/pia/Research/Hyphos.

[42]. Anith, C. (2008). An Overview of mobile agent in Mobile Computer, A term Paper. Department of Computer Science, Ebonyi State University, 32-66.
[43]. Milojicic, D. (1999). Mobile Agent Applications. IEEE Concurrency. *Journal of Personal Technologies*, (3), 80-90.
[44]. Sanneck, H., Berger, M. and Bauer, B. (2003). Application of Agent Technology to next Generation Wireless/Mobile Networks, 67-71.
[45]. Zhang, R., Abdelzaher, T. F. and Stankovic, J. A. (2004). Efficient TCP connection failover in web server clusters. In IEEE INFOCOM, 2, 1219-1228.

Ikporo, Stephen Chibueze (Ph.D), et. al. "Effective Network Traffic Management In A Heterogeneous Servers Cluster Using Hybridized Central And Mobile Agent- Based Approaches." *IOSR Journal of Mobile Computing & Application (IOSR-JMCA)*, *8*(5), (2021): pp. 09-20.