Received: 6 January 2025 | Revised: 3 July 2025 | Accepted: 11 August 2025 | Published online: 5 September 2025

RESEARCH ARTICLE

FinTech and Sustainable Innovation
2025, Vol. 1, A10, 12 Pages
DOI: 10.47852/bonviewFSI52025168

BON VIEW PUBLISHING

Distributed Harmony: Intelligent Load Balancing in Heterogeneous Server Clusters Using Mobile Agents

John Ugah[1], Steven Ikporo[1], Ifeanyi Achi[2] and Ugochukwu Onwudebelu[2,*]

[1]*Department of Computer Science, Ebonyi State University, Nigeria*

[2]*Department of Computer Science/Informatics, Alex Ekwueme Federal University Ndufu Alike, Nigeria*

Abstract: Intelligent network load balancing deals with efficiently distributing incoming network traffic among a collection of servers. A mobile agent-based load balancing approach leverages intelligent agents to identify underutilized servers within a cluster and dynamically distribute incoming workloads to optimize resource allocation. Intelligent load balancing is crucial because most available network load balancing techniques are currently in use such as round robin, weighted round robin, least connection, and weighted least connect, and many others suffer from delay in response time, computational overhead, increased bandwidth usage, and traffic collision, and lack of distribution intelligence. An intelligent approach to network load balancing presents an algorithm that brings about equitable distribution of network traffic loads in cluster networks. Intelligent mobile agents use intelligent information to identify and generate responses on the availability and status of each server within the cluster and in turn equitably distribute the incoming network traffic to the available servers intelligently. Data was gathered using interviews, questionnaires, observations, and reviews of related literature. The analysis of the questionnaires was done using SPSS software. To assess the goodness of fit, a chi-square test was employed at the inferential level, providing statistical validation of the results. An object-oriented methodology was adopted. We also used the following tools: Java, MySQL, NVivo, and C#. The result clearly indicates that intelligent network load balancing techniques drastically reduce the response time delay and computational overhead usually experienced with other network load balancing techniques in use. It also facilitates the utilization of resources by providing a throughput with minimum response time, sharing the workload equally between the available servers depending on their status. Network traffic and bandwidth consumption are considerably reduced. The overall effect is that Internet usage is made more user-friendly both for professionals and nonprofessionals, thereby bringing about a reduction in cost.

Keywords: network-traffic, mobile agent, load balancing, distributed harmony, heterogeneity, data center

1. Introduction

An open network in recent years has witnessed a tremendous increase in the number of servers with which it provides services and fault tolerance. According to Eludiora et al. [1], the services on the web have significantly increased such that more than six million users access the Internet daily. The proliferation of web services and user requests necessitates web servers that offer high availability, scalability, reliability, and efficiency to ensure rapid response times and high throughput, catering to the ever-increasing demand [2]. The users' requests to these servers are also increasing significantly every minute. Furthermore, the increasing rate of the World Wide Web and popular Web servers has led to the wide adoption of distributed web servers in the form of clusters to service the increasing number of client requests, as single servers can no longer handle the workloads again. The effective distribution of these loads among these servers to avoid congestion and denial of service is of great concern. Research by Dhas and Uthariaraj [3] highlights the importance of distributing incoming client requests among distributed web servers to enhance network performance. Network load balancing enhances the availability and scalability of Internet servers' applications such as Web, File Transfer Protocol (FTP), etc., by ensuring that workload is distributed among a cluster of backend servers [4]. This is achieved by moving some loads from the heavily loaded nodes to the lightly loaded nodes, thereby achieving better performance. Several hardware and software-based load balancing techniques are already available worldwide [5]. They include the Domain Network Server (DNS) scheduler approach of load balancing, Queuing Network Model, Dynamic Feedback Mechanism, etc. Load balancers continually monitor client requests and redirect them accordingly, while also periodically collecting server availability status to maintain up-to-date information. Servers provide services to clients and share their current load information with the load balancing algorithm. However, the communication overhead between servers and load balancers can hinder even workload distribution, particularly in open networks. To overcome this, there is a need for an intelligent agent (mobile agent)-based model, which will study the availability of the servers and also migrate loads from

***Corresponding author:** Ugochukwu Onwudebelu, Department of Computer Science/Informatics, Alex Ekwueme Federal University Ndufu Alike, Nigeria. Email: ugochukwu.onwudebelu@funai.edu.ng

the overloaded nodes to the less loaded nodes to avoid downtime, thereby making a real-time decision on which node will receive the migrating load. This model was developed using the Platform for Load Balancing (PLB) framework and implemented leveraging the Platform for Mobile Agent Distribution and Execution (PMADE), along with contemporary programming languages. Markov decision processes are commonly utilized to represent queuing systems and facilitate optimal control in both static and dynamic settings [6], where the agent relies on either immediate or delayed feedback.

2. Literature Review

This research involved an extensive review of various studies. Cardellini et al. [7] introduced the DNS scheduler approach to load balancing in multi-server environments. They noted that overloaded servers and high bandwidth utilization lead to increased waiting times for Internet access. However, the scheduler's control over job scheduling is limited, as once the Universal Resource Locator (URL) is translated to an IP address, it is cached in the client's browser, resulting in skewed server loads [8]. Li and Kameda [9] proposed a load balancing approach for multi-servers in distributed environments using job scheduling policies. This method assigns loads to servers based on partial load information. This results in high message overhead. The research by Kumar and Singh [10] was based on giving an overall need and architecture of a common load balancing system, which turned out to be useful to many applications (such as PayTM, MakeMyTrip, Amazon, and Flipkart) consuming different distributed computing over a single platform. Mobile agents have been researched by many authors like Tripathi et al. [11]. Cao et al. [12] developed a mobile agent-based load balancing approach for distributed web servers. This approach involves frequent message exchanges between agents and servers to detect and exchange load information, which increases bandwidth utilization. Pao and Chen [13], as well as Yu et al. [14], introduced a dispatcher-based web server load balancing architecture, where client requests are assigned to the server with the least load by the IP address dispatcher. However, this approach leads to high message complexity. Jaiswal and Jain [15] proposed a load balancing approach for multi-servers in distributed environments using job scheduling policies. This method assigns loads to servers based on partial load information, resulting in high message overhead.

Mobile agent-based load balancing system in heterogeneous server machines cluster comes to concatenate some of the approaches for better and efficient service delivery [16, 17]. The system ensures that there will be a central server whose duty is to initiate load distribution and transfer processes. The mobile agents will ensure that the server information is made available to the central/master server [18]. They are responsible for the load distribution to servers as well as server-to-server load transfer. Mobile agent-based load balancing also provides support for dynamic balancing of load across heterogeneous platforms and can carry all application-specific code with them, rather than requiring pre-installation of that code on the destination server. It also supported the disruptive nature of wireless links and alleviated its associated bandwidth limitations, thereby enhancing efficiency. Nehra et al. [19] presented a simulation model for partially observable systems that captures load balancing decisions in parallel buffered systems under limited information. The model uses a scalable Monte Carlo tree search algorithm to find a load balancing policy in real time. They were also able to demonstrate how a partial observability load balancer can optimize parallel processing in clusters by using real network data provided by Tahir et al. [20], as well as map incoming jobs to the parallel servers.

Al-Qerem et al. [21] proposed an adaptive transport protocol selection mechanism called WiseTrans, which switches transport protocols for mobile web services online to improve web request completion times. This innovation aimed to improve upon Quick UDP Internet Connections (QUIC), a transport protocol developed as a Transmission Control Protocol (TCP) alternative [22, 23]. WiseTrans leverages machine learning techniques to address temporal and spatial heterogeneity, enabling informed decision-making and online learning for optimal performance [21, 24]. Furthermore, WiseTrans could also be integrated with mechanisms that permit the selection of various variants of TCP [25]. Chen et al. [26] demonstrated the fact that emerging designs at the client side will affect the performance of TCP. In another study by Li et al. [27], researchers investigated the challenges of modeling edge-cloud computing environments and processing Internet of Things (IoT) transactions. Their proposed IoTT framework views IoT transactions as collections of fundamental elements. Simulation experiments evaluated the performance of their scheduling method and three transaction execution algorithms under network disconnection scenarios. The study compared three implementation strategies: Edge Host Strategy, Cloud Host Strategy, and Hybrid BHS, which execute IoT transactions on Edge hosts, cloud, and both, respectively.

Li et al. [27] proposed a novel intelligent mobile agent approach for cloud environments. This Java-based mobile agent detects underutilized and overloaded data center components, regulating server and switch activities to shut down underutilized components. This research addresses energy efficiency, cost-awareness, and system performance issues.

Various studies have investigated load balancing and resource optimization techniques. For instance, Charan [28] introduced a traffic-aware adaptive granularity load balancing design (TLB) that reduces flow completion time for short flows and improves throughputs for long flows. TLB dynamically adjusts the rerouting granularity of long flows based on the strength of the load of short flows. In another study, Hu et al. [29] examined the Join-Below-Threshold load balancing policy in a heterogeneous system of resource-sharing servers. The authors proved the convergence of stationary measures to the fixed point of the mean field model and derived sufficient conditions for the existence of a unique fixed point. Horváth et al. [30] formulated task scheduling for heterogeneous Hadoop YARN clusters as a constrained energy consumption optimization problem and proposed a heuristic algorithm based on load balancing and deadline constraints. The results showed improved completion time and energy consumption compared to alternative methods.

The SWARM protocol, introduced in Gao and Huang [31], continuously monitors workload across distributed machines in spatial data streaming systems. SWARM adjusts workload distribution to detect performance bottlenecks and handles multiple query-execution and data-persistence models. Daghistani et al. [32] presented Fast, Linearizable, Network-Accelerated Client Reads (FLAIR), a protocol that leverages programmable switches to accelerate read operations without affecting writes. FLAIR identifies consistent replicas and implements load balancing techniques to distribute the load. In Kettaneh et al. [33], the authors proposed HLB, a load-aware Layer-4 load balancer that estimates server occupancies and processing speeds based on networking observations. HLB offers improved load balancing performance and adaptability to dynamic environments. Yao et al. [34] introduced Reordering-Robust Load Balancing (RLB), which augments existing load balancing

algorithms to reduce packet reordering in lossless datacenter networks. RLB predicts PFC triggering and makes timely rerouting decisions. Hu et al. [35] studied load balancing optimization methods for in-memory databases in IoT environments. The results showed improved performance using the proposed algorithm. Sun et al. [36] proposed Polygon, a Content Delivery Network (CDN) server selection system that perceives multiple resource demands based on the QUIC protocol and anycast routing. Polygon identifies suitable CDN servers and establishes fast connections. In Zhou et al. [37], the researchers optimized edge server placement in WMAN environments to ensure load balancing and reduce network traffic. The proposed greedy algorithm showed improved performance compared to existing algorithms. Vali et al. [38] introduced Tarazu, an end-to-end control plane for optimizing parallel and distributed HPC I/O systems. Tarazu provides efficient load balancing among storage servers and manages performance degradation due to resource contention. These studies demonstrate various approaches to load balancing and resource optimization, highlighting the importance of efficient resource allocation in different environments (see Table 1).

3. Research Methodology

Our methodology describes data collection using questionnaires, interviews, and literature review. Our experimental setup consisted of a high-performance server running an Intel Core i7-12700K processor with 16GB RAM and a 1TB SSD. The software environment included Ubuntu 22.04 as the operating system, Python 3.10 for scripting, and SPSS for statistical analysis, as well as NVivo, Java, MySQL, and C#. Our system modeling includes using Unified Modeling Language (UML) diagrams. The mobile agent system was implemented using the PMADE and PLB frameworks.

The methods adopted in this research included aspects that determined the weaknesses of the existing load balancing techniques in use today. The method is similar to the most recent international network load balancing techniques. The first step was to carry out a detailed literature review of the past and recent work on load balancing. The second step was to identify the specific vulnerabilities of the existing load balancing techniques through the literature to

Table 1
Recent applications in enhancing load balancing

Reference/problem identified	Algorithm/technology used	Achievement made/brief discussion on result/research parameters	Shortcoming of the research for open research
Daghistani et al. [32]/Leader bottleneck for reads limits performance under read-heavy loads. This results in read performance bottlenecks: higher latency and lower throughput, especially underread heavy workloads.	Programmable switches (e.g., Barefoot Tofino) run a P4 pipeline that tracks client requests and replicas' reply state, etc. In-network packet inspection and forwarding logic using programmable switches. Implementation with FlairKV: P4 pipeline on Tofino, integrated with Raft KV-store	Throughput: Up to 42% higher compared to traditional forwarding-to-leader. Latency: 35%–97% lower, particularly under skewed and read-heavy workloads. Scenarios evaluated: Uniform, read-hot, and read-local access patterns show strong gains	Hardware needs, switch state overhead, protocol specificity, complex failover scenarios
Yao et al. [34]/Packet reordering in PFC-driven, lossless Content Delivery Networks (CDNs) due to unhandled PFC events	Reordering-Robust Load Balancing (RLB), Integration with Existing Load Balancing Mechanisms, PFC prediction via queue-length derivatives, and upstream notification, as well as in-network recirculation/reroute	Up to 72% improvement in tail flow-completion time across four load balancing schemes	Dependency on Programmable Hardware, Complexity of Implementation, Scalability Concerns, Overhead of In-Network Recirculation, Tuning complexity, hardware-specific, scalability/performance trade-offs, lossless-only context
Paul et al. [39]/The need to improve the operation effect of the in-memory database for massive information processing of the Internet of Things.	Biased Random Walk Search Algorithm, Adaptive indexing Techniques, Database Cracking Algorithms, Buffered-Swapping Algorithm, Rough index algorithms, consistent Hashing Algorithms, Redis DB, and Proxy Load Balancer	An adaptive FSST algorithm was developed. The author combines the load balancing signal processing algorithm to carry out the load balancing optimization analysis of the in-memory database. The results show that when the number of concurrent users is the same, the time consumption, throughput, and bandwidth of the developed methods are always higher than the method proposed in reference [40]	Limited real-world validation, insufficient performance metrics, overhead of adaptive indexing not discussed, security and privacy ignored, no clear trade-off analysis, as well as lack of details on hardware requirements

(*Continued*)

Table 1
(*Continued*)

Reference/problem identified	Algorithm/technology used	Achievement made/brief discussion on result/research parameters	Shortcoming of the research for open research
Sun et al. [36]/Mainstream Content Delivery Network (CDN) server selection schemes can only consider a single resource type and are unable to choose the most suitable servers when faced with diverse resource types.	Allocation algorithm, Web content, video streaming, and replicable databases. They used Ping to measure network delay, IGI/PTR to measure bandwidth, Google Cloud Platform, The Mininet is configured with 64 CPU cores and 187 GB of memory.	They developed Polygon, a QUIC-powered CDN server selection system that is aware of multiple resource demands rather than a single resource type, which is equipped with customizable transport parameters to enable the seamless handling of resource requirements in requests. They monitored 3 resource types: network delay, bandwidth, and CPU capability [41].	They were unable to explore the deployment strategy of dispatchers. They are also unable to implement Polygon over other transport layers Protocols as well as the need to test Polygon in a more complicated browsing scenarios, while considering webpage structure and browser loading behavior.
Zhou et al. [37]/Edge Server Placement (ESP) problem. The issue of increased latency within the traditional cloud computing paradigm is a result of big data.	Random, Top K, k-means, genetic algorithms, and using the Python programming language (Python 3.11), integer linear programming model, GAMS software environment using Barron's Solver.	They developed a novel Recursive clustering technique, RESP, for Edge Server Placement in MEC environments. The RESP operates based on the media of each cluster, which is strategically placing edge servers to achieve workload balance and minimize network traffic between them.	The accuracy of load estimation significantly influences the effectiveness of the RESP approach. Thus, regular updates and improvements in load estimation techniques are needed. Therefore, it is essential to improve the accuracy of load estimation, optimize threshold selection, and address heterogeneity in management servers.
Vali et al. [38]/The issue of imbalanced I/O load on large parallel file systems upsets the performance of high-performance computing (HPC) applications of the parallel I/O.	Load balancing techniques	They developed Tarazu, an end-to-end control plane where clients transparently and adaptively write to a set of selected I/O servers to achieve balanced data placement with improvement of up to 33% in load balance and 43% in read performance when compared to the state-of-the-art.	Their design has a performance bottleneck issue, as well as the need to improve the I/O latency for file creation and write requests.
Hu et al. [35]/Deployment of streaming applications is a challenging problem in distributed stream computing systems.	Artificial Bee Colony Strategy	They developed a performance-aware deployment framework for streaming applications in distributed systems/low latency and high throughput.	Lack of using real-time data monitoring, as well as lack of multiple data streams

be reviewed. The next step was extensive data gathering from three categories of people involved in the research.

1) The study area: The study area for this research included three categories of people: Internet service providers (ISPs), Network Administrators, and Network users.
2) The study population consisted of 50 chosen network administrators, 100 Internet users, and 20 ISPs across five cities: Abakaliki, Enugu, Abuja, Lagos, and Uyo. A total of 200 questionnaires were sent to all the participants by the researchers to help ascertain their experiences in the use of the current system.
3) Study inclusion: This research focused on different ISPs in Nigeria. The service providers involved were Yaklick, Galaxy Backbone, MTN, GLO, Airtel, Etisalat, CompuNet, etc. These service providers were selected because they are the major Internet providers in Nigeria. Their network administrators were also involved in order to gain an in-depth knowledge of operational issues and challenges in their delivery service to customers. Internet users were also included in the study to enable researchers to get first-hand information about their experiences from different network providers.

In the study design, the action research design was adopted in this study. At first, an exploratory approach was used to gain a better understanding of the problem. This was followed by designing and implementing intervention strategies, during which relevant observations were gathered. These interventions and observations were iterated cyclically until the problem was fully understood. The protocols start with conceptualizing and particularizing the problem. It further moved through several interventions and evaluations to arrive at the targeted solution. The rationale for adopting the action study design is the fact that action search studies often have direct and obvious relevance to practice. Action search design focuses on pragmatic and solution-driven research rather than testing theories only. Finally, the study design was adopted because it conforms to the general design principles. For data collection, several methods were used to collect data in this study. The instruments used were a questionnaire, interview, observation, and survey. A total of 200

questionnaires were sent to all the participants by the researchers to help ascertain their experiences in the use of the current system. The use of a questionnaire ensures confirmation and completeness of data and increases confidence in the credibility of our findings. Extensive reviews of relevant articles and related literature were carried out. The major stakeholders in each of the service providers used as a case study were interviewed. An interview was used to enable the researchers to have face-to-face interaction with stakeholders so as to extract their own experiences in network services/performance over the years. Observations of the happenings in some network service providers were also adopted by the researchers.

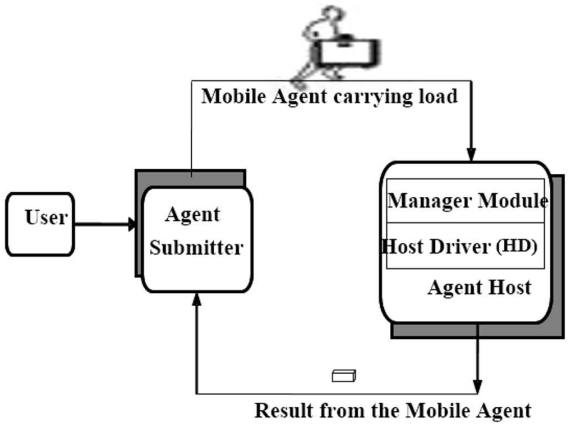
3.1. Data analysis and software development methods employed

The analysis of the questionnaires was done using SPSS software. The hypothesis was tested with the chi-square (X^2) test of goodness of fit at the inferential level. SPSS was used for descriptive statistics and data analysis, while chi-square tests were used to determine the significance of the results. The chi-square test was chosen because it is suitable for categorical data and can handle large sample sizes. The chi-square test was selected to assess the independence of task distribution patterns among servers, ensuring that the load balancing mechanism significantly differs from random assignment. SPSS was used for statistical validation due to its ability to handle large datasets and provide detailed analytical insights, including correlation and trend analysis. The interview responses were analyzed using Computer-Aided Quality Data Analysis Software (NVivo), which reshapes and restructures unstructured data. In modeling the new system, object-oriented methodology with a Rapid Application Development model of Software Development Life Cycle phases was adopted. UML, such as Use Case, Activity, sequence, and class diagrams, was used to develop the major decision scenarios regarding Internet users. The system was designed and implemented using the PLB framework, PMADE. Also, C# and Java were used for the interface design. My Structured Query Language (MySQL) was used to implement a database. The system was evaluated using five (5) server machines with the Microsoft Windows Server 2012 operating system. The research was conducted in accordance with TETFund's policy on research ethics, and appropriate measures were taken to protect the participants' confidentiality and anonymity.

3.2. Mobile agent design platform architecture

The Platform for Mobile Agent Development and Execution (PMADE) framework was utilized for designing mobile agents. The architecture comprises two primary components: the Agent Host (AH) and the Agent Submitter (AS). The AH executes incoming requests, while the AS submits requests to the AH on behalf of the user. The AH is the key component of PMADE architecture and consists of the manager modules and the Host Driver as shown in Figure 1.

Figure 1
Architecture of PMADE



3.3. Operational mechanism and implementation architecture

Mobile agents were structured using the PLB framework, which comprises three core components: Agents, Load Balancing Policy, and the Data Bank. The mobile agent utilizes content-based technology and a knowledge-based engine to assess server processing speed and availability. The implementation was based on the PMADE framework (Figure 2). The implementation leveraged the PMADE, as illustrated in Figure 1.

3.4. System design

UML, such as Use Case, Activity, and sequence and class, is used to develop the major decision scenarios. Figure 3 shows the use case diagram, while Figure 4 shows the sequence diagram of the mobile agent load balancing.

The sequence diagram of the mobile agent load balancing system is as shown in Figure 4.

The research was conducted in accordance with the ethical standards of the responsible committee on human experimentation. Ebonyi State University Policy on Research Ethics was followed. Written consent was sought through the directorate of research, commercialization, and innovation of Ebonyi State University, Abakaliki, and was secured. Also, all necessary measures were taken to protect all participants' confidentiality and anonymity.

Figure 2
Architecture of PLB

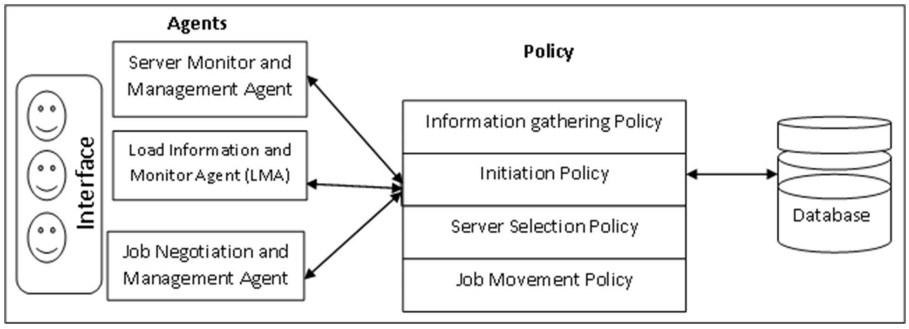


Figure 3
The use case diagram of mobile agent load balancing

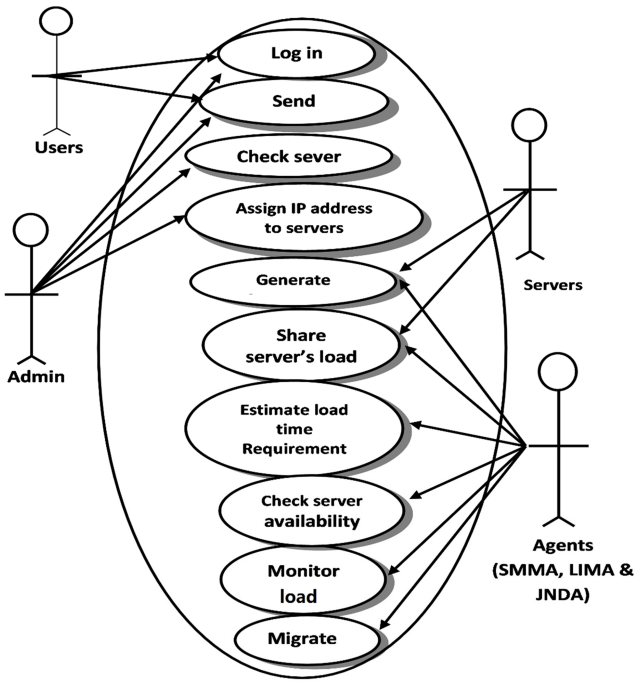
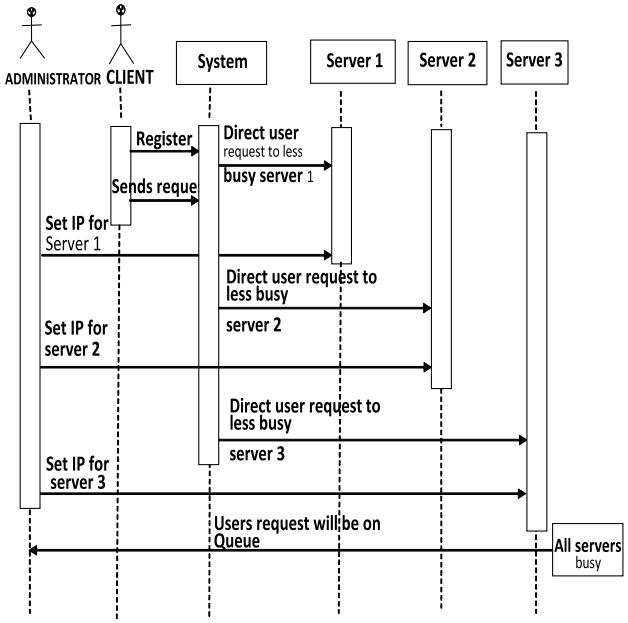


Figure 4
The sequence diagram of the mobile agent load balancing



4. Results and Discussion

4.1. Results

The problem that necessitated this study was the fact that network load balancing techniques such as round robin and centralized and other load balancing techniques that are currently being used are associated with some vulnerability. Some of these weaknesses include delay in response time, computational overhead, increased bandwidth usage, denial of service, load scheduling problem, traffic collision, and lack of distribution intelligence. This research kicked off by seeking answers to the following questions:

1) How would the introduction of mobile agents into the network load balancing help in addressing the problem of traffic congestion on the network?
2) How would the introduction of mobile agents help to improve the overall Internet service delivery to Internet users?

The following results were obtained at the end of the study: Two hundred (200) questionnaires were shared out of which 198 responses were collected. The responses indicated that 145 of the

Figure 5
Awareness of load balancing techniques in use

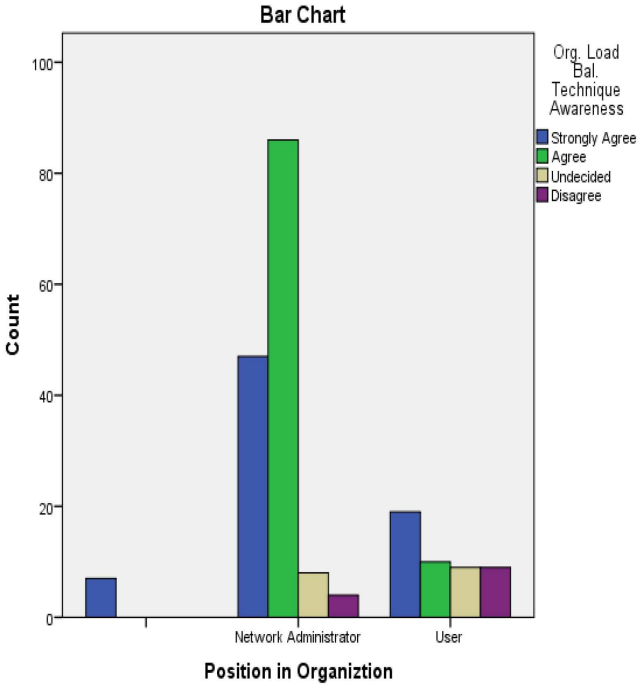
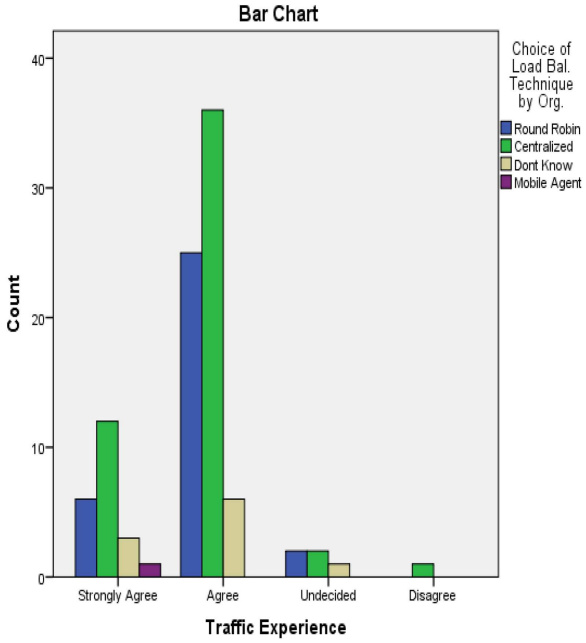


Figure 6
Network experiences of network administrators



respondents were network administrators. Out of this number, 45 of them strongly agreed that they were aware of the network load balancing techniques their company is using, 86 agreed that they were aware of the load balancing techniques of their organization, 8 of them were undecided, while 4 disagreed. The chart in Figure 5 illustrates responses.

Most ISPs agree that their company uses mostly round robin or centralized load balancing technique in handling loads. Figure 6 illustrates their responses.

ISPs and network administrators acknowledge that round robin and centralized and other techniques being used currently are associated with the problem of delay in response time, load scheduling problems, traffic collision, and lack of distribution intelligence. Figure 7 illustrates responses received from ISPs:

The introduction of mobile agents into the network load balancing helped in addressing the problem of traffic congestion and also helped to improve the overall Internet service delivery to Internet users. Figure 8 illustrates these findings.

Tables 2, 3, 4, and 5 show some results of the chi-square test for choice of load balancing techniques.

4.2. Discussion

The responses from the questionnaires distributed to respondents showed vividly that the majority of the network administrators are fully aware of the networking load balancing technique that their organization is using. The implicit techniques would enable them to detect when high traffic failure or congestion occurs due to the load balancing technique adopted by the organization. This would give the administrators a very good opportunity to provide a proven solution to the problem of traffic congestion.. This outcome was quite expected because network administrators should be the managers of every issue relating to load balancing in network services. The next outcome showed that most ISPs use centralized load balancing techniques in their organizations. This result actually agrees with the literature reviewed: The outcome was quite expected because most literature reviewed showed that major research efforts in this domain focused on a central approach.

Furthermore, ISPs and network administrators acknowledge that round robin and centralized and other techniques being used currently are associated with the problem of delay in response time load scheduling problem, traffic collision, and lack of distribution intelligence. This outcome agrees with the motivation for this study. The outcome reinforces the statement of the problem that guided this study. The statement of the problem was that network load balancing techniques such as round robin and centralized and other load balancing techniques that are currently being used are associated with the problem of delay in response time, computational overhead, increased bandwidth usage, denial of service, load scheduling problems, traffic collision, and lack of distribution intelligence. This result was actually expected.

The introduction of intelligent agent into the network load balancing helped in addressing the problem of traffic congestion and also helped to improve the overall Internet service delivery to Internet users. The responses from our respondents showed that 82 strongly agreed that there is a need for special network load balancing techniques and that they are also ready to adopt mobile agent-based load balancing. Similarly, 79 responses agreed with the fact that there is a need for a special mechanism for load balancing, 19 respondents were undecided, and 19 respondents

Figure 7
Causes of delay in delivery service

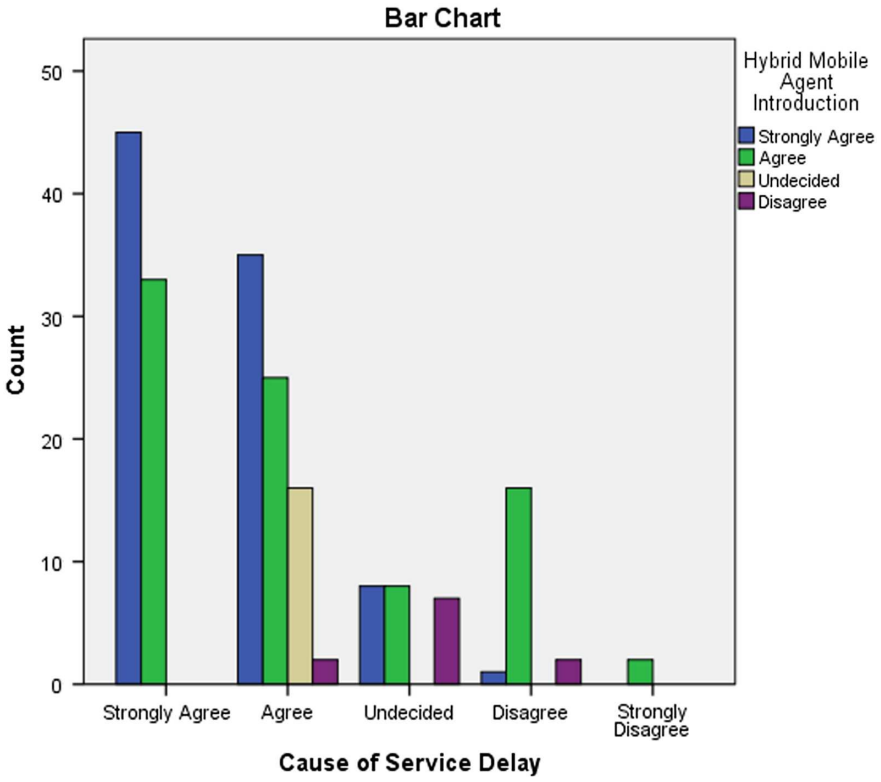


Figure 8
Introduction of mobile agent

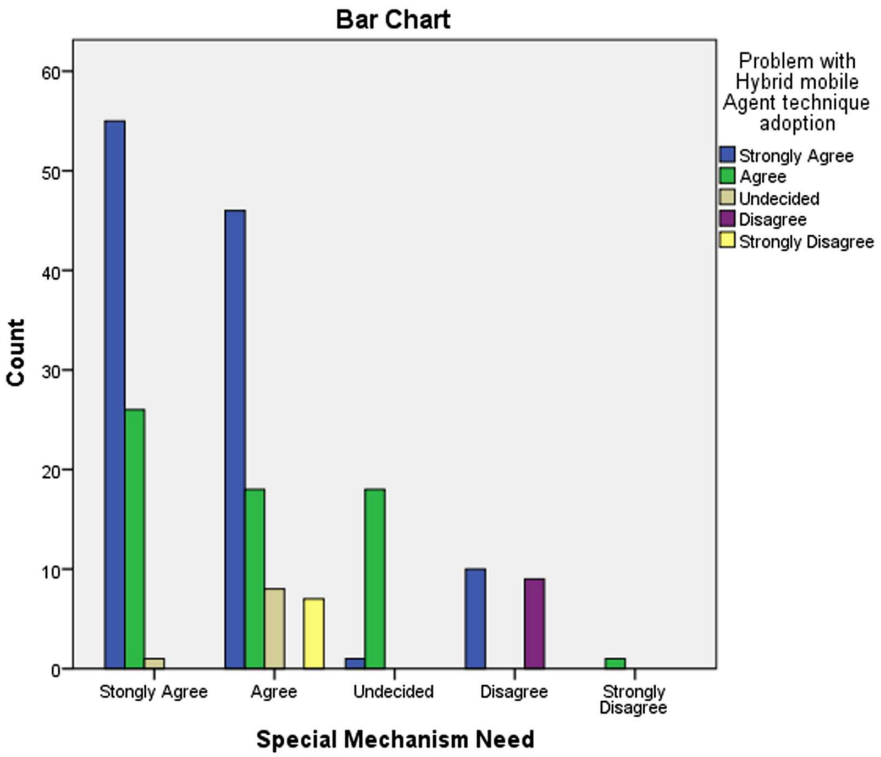


Table 2
Case processing table for choice of load balancing techniques

Case Processing Summary						
	Cases					
	Valid		Missing		Total	
	N	Percent	N	Percent	N	Percent
Traffic Experience* Choice of Load Bal. Technique by Org.	195	195.0%	5	5.0%	200	100.0%

Table 3
Analysis table for choice of load balancing techniques

Traffic Experience * Choice of Load Bal. Technique by Org. Crosstabulation							
			Choice of Load Bal. Technique by Org.				
			Round Robin	Central-ized	Don't Know	Mobile Agent	Total
Traffic Experience	Strongly Agree	Count	6	12	3	1	22
		Expected count	7.6	11.8	2.3	0.2	22.0
	Agree	Count	25	36	6	0	67
		Expected count	23.3	36.0	7.1	0.7	67.0
	Undecided	Count	2	2	1	0	5
		Expected count	1.7	2.7	0.5	0.1	5.0
	Disagree	Count	0	1	0	0	1
		Expected count	0.3	0.5	0.1	0.0	1.0
Total		Count	33	51	10	1	95
		Expected count	33.0	51.0	10.0	1.0	95.0

disagreed, while only 1 strongly disagreed. The implication of this result is that the current load balancing techniques being deployed by most ISPs are no longer very effective. It further showed that both ISPs and Internet users are yarning for more effective load balancing techniques. Respondents also indicated their readiness to adopt any new load balancing techniques that will enhance load

Table 4
Chi-square test for choice of load balancing techniques

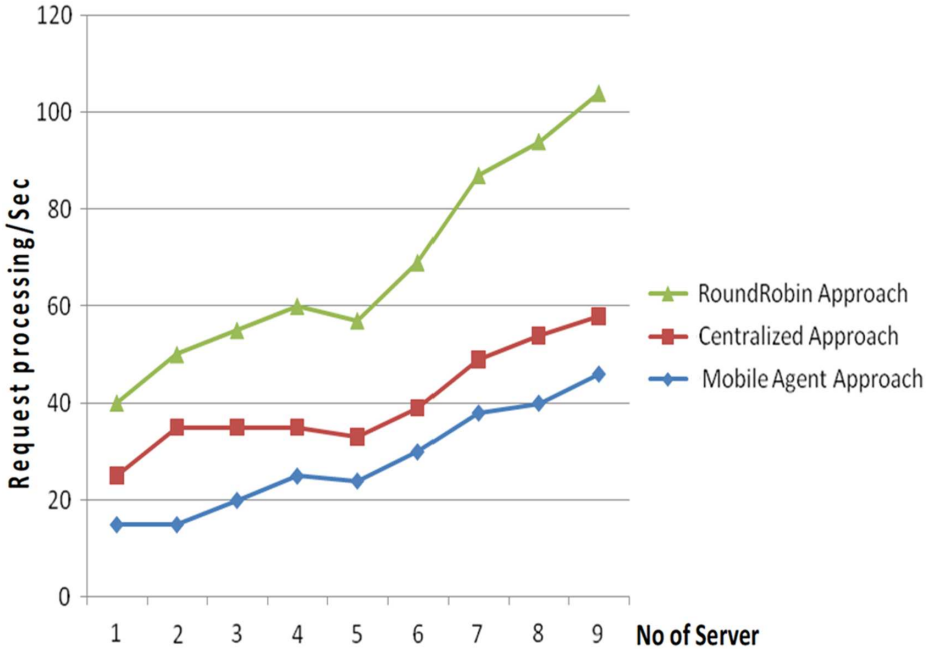
Chi-Square Tests			
	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	5.654[a]	9	0.774
Likelihood Ratio	5.574	9	0.782
N of Valid Cases	195		

Note: [a]11 cells (68.8%) have an expected count of less than 5. The minimum expected count is 0.01.

Table 5
Data analysis for introduction of mobile agent in load balancing

Cause of service Delay* Hybrid Mobile Agent Introduction Cross tabulation							
			Hybrid Mobile Agent Introduction				
			Strongly agree	Agree	Undecided	Disagree	Total
Cause of Service Delay	Strongly Agree	Count	45	33	0	0	78
		Expected count	34.7	32.8	6.2	4.3	78.0
	Agree	Count	35	25	16	2	78
		Expected count	34.7	32.8	6.2	4.3	78.0
	Undecided	Count	8	8	0	7	23
		Expected count	10.2	9.7	1.8	1.3	23.0
	Disagree	Count	1	16	0	2	19
		Expected count	8.5	8.0	1.5	1.0	19.0
	Strongly Disagree	Count	0	2	0	0	2
		Expected count	0.9	0.8	0.2	0.1	2.0
Total		Count	89	84	16	11	200
		Expected count	89.0	84.0	16.0	11.0	200.0

Figure 9
Comparison between round robin and centralized and mobile agent



balancing in heterogeneous server networks. The overall outcome of this result is a strong support for the introduction of intelligent agents (mobile agent approach) in traffic load balancing techniques so as to bring about more efficiency and effectiveness in load balancing techniques. The system developed was evaluated. The system's operational efficiency, effectiveness, and suitability were evaluated by simulating four servers. The system consists of a cluster of four servers, each with a quad-core processor, 16 GB RAM, and 1 TB storage. Mobile agents are used to distribute tasks across the servers. For the hardware specifications, the servers are equipped with Intel Xeon processors, 16 GB RAM, and 1 TB storage. One server out of the four servers served as the manager server, while the other three formed the cluster. The simulation showed that loads were distributed to the clustered servers by the manager server using load migration agents. Figure 9 captures the summary of the comparison between round robin and centralized and mobile agent load balancing techniques.

From Figure 9, it could be seen that in round robin, the amount of time it takes to process user requests increases with an increase in load. This was because round robin technique does not necessarily possess inbuilt intelligence that could help in distributing user requests as they increase. Looking at the diagram, it could be seen that in centralized techniques, the amount of time it takes in handling user requests even with an increase in no of servers is better when compared with that of round robin. However, centralized techniques do not give maximum results. Mobile agent techniques give more efficient and effective results. In mobile agent techniques, the addition of servers to the system and an increase in user requests do not in any way slow down the system. The reason is that it has inbuilt intelligence that helps the servers to distribute incoming loads evenly to the available servers, thereby not letting any server be overloaded. The performance evaluation of the system was measured by the efficiency in load distribution, and system throughput was optimal. The expected result was achieved and delivered. The efficiency in load distribution of each server was measured by checking the length of job in a queue for each server at different times. The throughput of the system was measured by the total number of requests processed per second. When the throughput of the traditional approach to load balancing and mobile agent approaches were compared, the mobile agent approach was found to be better than traditional. Further comparison of the network traffic between mobile agent and traditional approaches clearly shows that mobile agent approaches generate lower communication delays than the traditional approaches. This showed that there are efficient traffic handling and a reduction in bandwidth consumption and response time in the mobile agent approach.

5. Conclusion

In this research, a mobile agent approach to traffic load balancing in heterogeneous cluster server networks was developed. The system developed could be deployed by service providers such as ISPs. Some of the basic outcomes of this research include: (i) Design of a knowledge-based repository that calculates and stores the processing time of each load in the server. (ii) Development of a mobile agent-based load balancing system with feedback module for responses on server status, which uses the responses to migrate loads from overloaded server to lesser loaded or idle server where necessary. (iii) Comparison of a performance of the traditional approaches to traffic load balancing techniques to the mobile agent load balancing technique developed. In all, the mobile agent approach to traffic load balancing was seen as an efficient approach, and it drastically reduced the response time delay and computational overhead usually experienced with the existing system. Reduction of bandwidth usage, denial of service and service failure, or packet loss was also observed. Despite its efficiency, the proposed method has certain limitations. One limitation of the proposed approach is its scalability, as it may not perform well in very large-scale systems. Under extremely high workloads, computational overhead increases due to agent decision-making complexity. Additionally, the system may experience latency issues in a distributed cloud environment where network delays impact agent communication. Future improvements will focus on optimizing agent decision speed and incorporating predictive analytics to anticipate load spikes. Future work will also explore the integration of reinforcement learning to enhance agent decision-making in dynamic environments. Additionally, we plan to extend the model for cloud-native architectures using Kubernetes-based containerized deployment. Another direction involves reducing computational overhead by implementing lightweight agent frameworks optimized for edge computing scenarios.

Ethical Statement

This study received ethical approval from the University Research Ethics Committee (UREC) at Ebonyi State University, under approval number EBSU/DRIC/UREC/Vol. 05/038. All procedures involving human participants were conducted in accordance with the ethical standards and guidelines established by the committee. Informed consent was obtained from all participants prior to data collection, and no personally identifiable information was disclosed in the study.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support this work are available upon reasonable request from the corresponding author.

Author Contribution Statement

John Ugah: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Steven Ikporo:** Software, Investigation, Data curation. **Ifeanyi Achi:** Software, Formal analysis, Investigation, Data curation. **Ugochukwu Onwudebelu:** Methodology, Formal analysis, Resources, Writing – original draft, Writing – review & editing, Visualization.

References

[1] Eludiora, S., Abiona, O., Aderounmu, G., Oluwatope, A., Onime, C., & Kehinde, L. (2010). A load balancing policy for distributed web service. *International Journal of Communications, Network and System Sciences*, *3*(8), 645–654. https://doi.org/10.4236/ijcns.2010.38087

[2] Julka, R., & Kaur, H. (2014). Load balancing techniques using mobile agents. *IPASJ International Journal of Computer Science*, *2*(12), 34–38.

[3] Dhas, V. G., & Uthariaraj, V. R. (2014). Distribution of load using mobile agent in distributed web servers. *American*

Journal of Applied Sciences, *11*(5), 811–817. https://doi.org/10.3844/ajassp.2014.811.817

[4] Radojević, B., & Žagar, M. (2011). Analysis of issues with load balancing algorithms in hosted (cloud) environments. In *2011 Proceedings of the 34th International Convention*, 416–420.

[5] Tyagi, V., Singh, S., Wu, H., & Gill, S. S. (2024). Load balancing in SDN-enabled WSNs toward 6G IoE: Partial cluster migration approach. *IEEE Internet of Things Journal*, *11*(18), 29557–29568. https://doi.org/10.1109/JIOT.2024.3402266

[6] Ayesta, U. (2022). Reinforcement learning in queues. *Queueing Systems*, *100*(3), 497–499. https://doi.org/10.1007/s11134-022-09844-w

[7] Cardellini, V., Casalicchio, E., Colajanni, M., & Yu, P. S. (2002). The state of the art in locally distributed web-server systems. *ACM Computing Surveys*, *34*(2), 263–311. https://doi.org/10.1145/508352.508355

[8] Jafri, S. A. R., Hong, Y.-J., Thottethodi, M., & Vijaykumar, T. N. (2010). Adaptive flow control for robust performance and energy. In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 433–444. https://doi.org/10.1109/MICRO.2010.48

[9] Li, J., & Kameda, H. (1998). Load balancing problems for multiclass jobs in distributed/parallel computer systems. *IEEE Transactions on Computers*, *47*(3), 322–332. https://doi.org/10.1109/12.660168

[10] Kumar, A., & Singh, D. (2021). Artificial intelligent load balance agent on network traffic across multiple heterogeneous distributed computing systems. *SSRN*. https://doi.org/10.2139/ssrn.3739322

[11] Tripathi, A., Ahmed, T., Pathak, S., Carney, M., & 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' (Cat. No. 02CH37327)*, 65–78. https://doi.org/10.1109/NOMS.2002.1015546

[12] Cao, J., Sun, Y., Wang, X., & Das, S. K. (2003). Scalable load balancing on distributed web servers using mobile agents. *Journal of Parallel and Distributed Computing*, *63*(10), 996–1005. https://doi.org/10.1016/S0743-7315(03)00099-6

[13] Pao, T.-L., & Chen, J.-B. (2006). The scalability of heterogeneous dispatcher-based web server load balancing architecture. In *2006 Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies*, 213–216. https://doi.org/10.1109/PDCAT.2006.110

[14] Yu, Y. W.-Y., Tsai, S.-J., Liou, Y.-J., Hong, C.-J., & Chen, T.-J. (2006). Association study of two *serotonin 1A receptor* gene polymorphisms and fluoxetine treatment response in Chinese major depressive disorders. *European Neuropsychopharmacology*, *16*(7), 498–503. https://doi.org/10.1016/j.euroneuro.2005.12.004

[15] Jaiswal, A. A., & Jain, R. (2016). Dynamic load management in cloud computing environment. *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, *4*(9), 186–190.

[16] Cao, J., & Das. S. K. (Eds.). (2012). *Mobile agents in networking and distributed computing*. USA: Wiley. https://doi.org/10.1002/9781118135617

[17] Rao, P. C. S., Jana, P. K., & Banka, H. (2017). A particle swarm optimization based energy efficient cluster head selection algorithm for wireless sensor networks. *Wireless Networks*, *23*(7), 2005–2020. https://doi.org/10.1007/s11276-016-1270-7

[18] Tyagi, V., & Singh, S. (2023). GM-WOA: A hybrid energy efficient cluster routing technique for SDN-enabled WSNs. *The Journal of Supercomputing*, *79*(13), 14894–14922. https://doi.org/10.1007/s11227-023-05263-7

[19] Nehra, N., Patel, R. B., & Bhat, V. K. (2007). A framework for distributed dynamic load balancing in heterogeneous cluster. *Journal of Computer Science*, *3*(1), 14–24. https://doi.org/10.3844/JCSSP.2007.14.24

[20] Tahir, A., Alt, B., Rizk, A., & Koeppl, H. (2023). Load balancing in compute clusters with delayed feedback. *IEEE Transactions on Computers*, *72*(6), 1610–1622. https://doi.org/10.1109/TC.2022.3215907

[21] Al-Qerem, A., Ali, A. M., Nashwan, S., Alauthman, M., Hamarsheh, A., Nabot, A., & Jibreen, I. (2023). Transactional services for concurrent mobile agents over edge/cloud computing-assisted social internet of things. *Journal of Data and Information Quality*, *15*(3), 36. https://doi.org/10.1145/3603714

[22] Zhang, J., Ren, S., Dong, E., Meng, Z., Yang, Y., Xu, M. Yang S., ..., & Yue Y. (2023). Reducing mobile web latency through adaptively selecting transport protocol. *IEEE/ACM Transactions on Networking*, *31*(5), 2162–2177. https://doi.org/10.1109/TNET.2023.3235907

[23] Iyengar, J., & Thomson, M. (2021). *QUIC: A UDP-based multiplexed and secure transport*. Retrieved from: https://rfc-editor.org/rfc/rfc9000.txt

[24] Yu, A., & Benson, T. A. (2021). Dissecting performance of production QUIC. In *Proceedings of the Web Conference 2021*, 1157–1168. https://doi.org/10.1145/3442381.3450103

[25] Abbasloo, S., Yen, C.-Y., & Chao, H. J. (2020). Classic meets modern: A pragmatic learning-based congestion control for the internet. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, 632–647. https://doi.org/10.1145/3387514.3405892

[26] Chen, K., Shan, D., Luo, X., Zhang, T., Yang, Y., & Ren, F. (2020). One rein to rule them all: A framework for datacenter-to-user congestion control. In *Proceedings of the 4th Asia-Pacific Workshop on Networking*, 44–51. https://doi.org/10.1145/3411029.3411036

[27] Li, T., Zheng, K., Xu, K., Jadhav, R. A., Xiong, T., Winstein, K., & Tan, K. (2020). TACK: Improving wireless transport performance by taming acknowledgments. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, 15–30. https://doi.org/10.1145/3387514.3405850

[28] Charan, C. S. (2022). Improving energy efficiency in cloud computing data centres using intelligent mobile agents. *International Journal of Engineering Technology and Management Sciences*, *5*(6), 748–752. https://doi.org/10.46647/ijetms.2022.v06i05.116

[29] Hu, J., Huang, J., Lyu, W., Li, W., Li, Z., Jiang, W., . . . , & He, T. (2021). Adjusting switching granularity of load balancing for heterogeneous datacenter traffic. *IEEE/ACM Transactions on Networking*, *29*(5), 2367–2384. https://doi.org/10.1109/TNET.2021.3088276

[30] Horváth, I. A., Scully, Z., & van Houdt, B. (2020). Mean field analysis of join-below-threshold load balancing for resource sharing servers. *ACM SIGMETRICS Performance*

Evaluation Review, *48*(1), 41–42. https://doi.org/10.1145/3410048.3410072

[31] Gao, Y., & Huang, C. (2021). Energy-efficient scheduling of MapReduce tasks based on load balancing and deadline constraint in heterogeneous hadoop YARN cluster. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design*, 220–225. https://doi.org/10.1109/CSCWD49262.2021.9437771

[32] Daghistani, A., Aref, W. G., Ghafoor, A., & Mahmood, A. R. (2021). SWARM: Adaptive load balancing in distributed streaming systems for big spatial data. *ACM Transactions on Spatial Algorithms and Systems*, *7*(3), 14. https://doi.org/10.1145/3460013

[33] Kettaneh, I., Alquraan, A., Takruri, H., Mashtizadeh, A. J., & Al-Kiswany, S. (2022). Accelerating reads with in-network consistency-aware load balancing. *IEEE/ACM Transactions on Networking*, *30*(3), 954–968. https://doi.org/10.1109/TNET.2021.3126203

[34] Yao, Z., Desmouceaux, Y., Cordero-Fuertes, J.-A., Townsley, M., & Clausen, T. (2022). HLB: Toward load-aware load balancing. *IEEE/ACM Transactions on Networking*, *30*(6), 2658–2673. https://doi.org/10.1109/TNET.2022.3177163

[35] Hu, J., He, Y., Luo, W., Huang, J., & Wang, J. (2024). Enhancing load balancing with in-network recirculation to prevent packet reordering in lossless data centers. *IEEE/ACM Transactions on Networking*, *32*(5), 4114–4127. https://doi.org/10.1109/TNET.2024.3403671

[36] Sun, D., Gao, S., Liu, X., Li, F., & Buyya, R. (2020). Performance-aware deployment of streaming applications in distributed stream computing systems. *International Journal of Bio-Inspired Computation*, *15*(1), 52–62. https://doi.org/10.1504/IJBIC.2020.105892

[37] Zhou, M., Guo, T., Chen, Y., Wan, J., & Wang, X. (2021). Polygon: A QUIC-based CDN server selection system supporting multiple resource demands. In *Proceedings of the 22nd International Middleware Conference: Industrial Track*, 16–22. https://doi.org/10.1145/3491084.3491428

[38] Vali, A. A., Azizi, S., & Shojafar, M. (2024). RESP: A recursive clustering approach for edge server placement in mobile edge computing. *ACM Transactions on Internet Technology*, *24*(3), 13. https://doi.org/10.1145/3666091

[39] Paul, A. K., Neuwirth, S., Wadhwa, B., Wang, F., Oral, S., & Butt, A. R. (2024). Tarazu: An adaptive end-to-end I/O load-balancing framework for large-scale parallel file systems. *ACM Transactions on Storage*, *20*(2), 11. https://doi.org/10.1145/3641885

[40] Xu, H. (2024). In-memory database load balancing optimization for massive information processing of the Internet of Things. *ACM Transactions on Asian and Low-Resource Language Information Processing*. Advance online publication. https://doi.org/10.1145/3670996

[41] Cerroni, W., Galis, A., Shiomoto, K., & Zhani, M. F. (2019). Telecom software, network virtualization, and software defined networks. *IEEE Communications Magazine*, *57*(10), 40–41. https://doi.org/10.1109/MCOM.2019.8875711

How to Cite: Ugah, J., Ikporo, S., Achi, I., & Onwudebelu, U. (2025). Distributed Harmony: Intelligent Load Balancing in Heterogeneous Server Clusters Using Mobile Agents. *FinTech and Sustainable Innovation*, *1*, A10. https://doi.org/10.47852/bonviewFSI52025168