A LOAD BALANCING SCHEME FOR DYNAMIC E-VOTING SYSTEM USING MOBILE AGENTS

[1, a, *]Ibharalu, F. T. and [2, b]Hammed M.

[1]Department of Computer Science, Federal University of Agriculture Abeokuta, Ogun, State, Nigeria

[2]Department of Computer Science, School of Applied Science, Federal Polytechnic, Ilaro, Ogun State, Nigeria

[a]tombharalu@yahoo.com (+2348033447288) [b]tundemuhammedy2k@yahoo.com (+2349069457169)

Abstract

Dynamic electronic voting system is a voting system that accommodates remote voters to cast their votes irrespective of where they are. It is a potential solution for voter low turnout at the polling station. However, it faces many challenges for maintaining accuracy and consistency of the voting process. One of the challenges is that a dynamic e-voting system must be able to handle the traffic, or else system performance will be degraded. The dynamic e-voting system need to have a quick response time. In a multi-server environment, if a server is heavily loaded, to achieve a quick response time will be difficult. This work proposed a load balancing scheme for dynamic e-voting system using mobile agent. This approach promises improved efficiency and effectiveness in terms of availability of the system for large-scale election.

Keywords: Dynamic e-voting, load balancing scheme, mobile agent, Hungarian method.

Definition of terms used in this study:

i. s = number of servers (parallel service channel, in queuing system)

ii. λ_n = Mean arrival rate (expected number of voters arriving per unit time) of new voters when n voters are in system

iii. $\frac{1}{\mu}$ = Expected service time

iv. λ is a constant for all n,

v. $\frac{1}{\lambda}$ = Expected inter-arrival rate

vi. μ_n Mean service rate for overall system (expected number of voters completing service per unit time) when voters are in system

vii. $\mu_n = s\mu$ when $n \geq s$ (all servers are busy)

viii. $p = \lambda(s\mu)$ is the utilization factor for the service facility, i.e., the expected fraction of time the individual servers are busy

ix. $p = \lambda(s\mu)$ is the utilization factor for the service facility, i.e., the expected fraction of time the individual servers are busy.

x. p_n is Probability of exact n voters in queuing system

xi. L = expected number of voters in queuing system = $\sum_{0}^{\infty} p_n$

xii. L_q = Expected queue length (excludes voters being served) $\sum_{n=0}^{\infty} (n-s)\,\mathrm{p_n}$,

xiii. W = waiting time in system (include service time) for each voters

xiv. M = Number of servers to be visited by agent

xv. $M-1$ Possible ways of agent tour

* Corresponding author: Ibharalu, F. T., Email: tomibharalu@yahoo.com, Tell: +2348033447288

xvi. d_q Distance between virtual server (i) and subordinate server/local server (j)

xvii. C_{ij} = cost of travelling between server (i) and server (j)

xviii. T_{ij} = Time to taken travel between server (i) and server (j)

xix. x_{ij} = Assignment of load to server (i) and server (j)

1.0 **Introduction**

Dynamic e-voting system accommodates system accessibility from voters' personal computers or even public computers; this could be a suitable way for many people that want to vote. It could also be a potential a solution to the low voter turnout at the polls [1]. However, dynamic e-voting system is a distributed computing that faced many challenges for maintaining accuracy/currency of the voting process [2]. The challenge is that the remote internet voting system must be able to handle the traffic, or else the performance will be degraded. If voters are frustrated with processing time, they will abandon the system and they may not have to vote again [1]. Voters' waiting in queue is possible when many voters want to authenticate and cast their votes and the system processing time is very slow. The length of time waiting to vote has regularly been an issue in the voting for the past decade. Voter's waiting time is widely accepted to be a major impediment to voter's turnout at election [3]. System availability is the key for having a successful remote voting system. Availability includes sufficient server processing capacity to handle steady traffic loads as well as large peaks that might occur during different times of the day [1]. This study proposed a load balancing techniques using mobile agents to ensure that the system is balanced, this enhances system response time and minimized voter's waiting time.

2.0. **Literature Review**

Many researchers have worked on dynamic e-voting. But most of these works focused on security aspects and other aspects such as resources accessibility, system availability, scalability etc. were not taken into cognizance. This research focuses on resources accessibility and system availability for large-scale election.

2.1 **Load Balancing and Distribution**

Load balancing plays an important role in system efficiency improvement. The two main types of load balancing algorithm: *static load balancing algorithm* and *dynamic load balancing algorithm.*

Static Load Balancing Algorithm

In this method, the performance of the nodes is determined at the beginning of execution. Then depending upon their performance the workload is distributed in the start by the master node. The slave processors calculate their allocated workloads and submit the results to the master. A task is always executed on the node to which it is assigned. This is a static load balancing method and is non-pre-emptive. Major static load balancing algorithms are Round Robin Randomized, Central Manager Algorithm and Threshold Algorithm. [4]. A major attribute of static schemes is that the final selection of a host for process allocation is made when the process is created and changes in the system cannot be made during the process execution. [5].

Dynamic Load Balancing Algorithm

Dynamic Load Balancing Algorithm: dynamic algorithms have the potential to outperform static algorithms by using *system-state* information to improve the quality of their decisions. Unlike static algorithms, dynamic algorithms allocate processes dynamically when one of the processors becomes under loaded. Instead, they are buffered in the queue on the main host and allocated dynamically upon requests from remote hosts, [5]. This method is consisted of Central Queue Algorithm and Local Queue Algorithm.[6].

2.2 **Mobile Agent**

A software agent is a piece of software that acts for a user or other program in a relationship of agency. Such "action on behalf of" implies the authority to decide which (and if) action is appropriate. The idea is that agents are not strictly invoked for a task, but activate themselves [7]. The mobile agent technology which includes at least one of the following features: data collection, searching and filtering, distribution and monitoring, information dissemination, negotiating and parallel processing [8]. Mobile agent can also summarize the load balancing polices and travel to another node and it can make a decision on load distribution according to the latest state [9]. It also has capability to reduce the network traffic, provide greater flexibility, overcome network latency and also reduce network load [9]. Many researchers have proposed number of load balancing in heterogeneous web server system based on dynamic load balancing using mobile agents [4, 7, 8, 10]. They proposed three components to design dynamic load balancing mechanism namely, the server module agent (SMA), the load information agent (LIA), and the job dispatching agent (JDA). In the dynamic load balancing proposed by the researchers mentioned above, the state information is gathered at frequent intervals and used to make decisions. This information exchange takes place in the form of messages which are exchanged. The communication delays in these exchanges result in uncertainty in the state information which is used to make decisions [10]. The advantage of this method is that the time required to complete a task by using mobile agents in coordinated fashion for dynamic load sharing mechanism is significantly less. Due to delays during the communication/transferring of load between (SMA) and recipient servers, it was considered that the information gathered by the mobile agent about a particular server may no longer accurately represent the current state of that server after these delays. The communication/transfer of load delays is represented using equation [11].

$$J_i(t_1, t_2) = \sum_{k:t_1<T_k\le t_2} Q_k \qquad (1)$$

The counting process $J_i(t_1, t_2)$ denotes the number of such external tasks arriving at node i in the interval process (t_1, t_2). While T_k are the arrival times of task requests, these task request follow the Poisson process with rate, λ_i and $Q_{k\,(k=1,2,\ldots,n)}$ is an integer-valued random variable associated with k^{th} task request.

3.0 **Related Work**

Several approaches have been proposed to deal with various problems in e-voting system. Discrete event simulation method is a client-server approach [12]. The greedy improvement in this algorithm generates reasonable allocations of voting resources, but in heterogeneous polling stations, which have different voter-arrival patterns and different turnout rates, the different distributions of voting times are difficult to control. Therefore, the voter's waiting time is increases due to slow system response time. This system cannot be implemented for large-scale election. In a generic model for performance of internet voting system proposed by [1], the system security requirements and traffic workload was completely well handled, but e-voting agent system is the best [13]. Hence, the time required to complete a task by using mobile agents in coordinated fashion is less, this will improve the system throughput and system response time. The Queuing Model as a Technique of Queue Solution in Nigeria Banking Industry proposed by [14], the authors observed that, in a situation where facilities are limited and cannot satisfy the demand made upon them, bottlenecks occur which manifest as queue (but customers are not interested in waiting in queues). When customers wait in queue, there is the danger that waiting time will become excessive leading to the loss of some customers to competitors. The author considered that the use of four-server system, eliminates waiting time, but at a higher cost which is not optimal too. Consequent upon this, a three-server model to reduce total expected costs and increase customer satisfaction was recommended. Three-server model is not efficient and effective for large-scale implementation (i.e. when there is increasing in the number of voters) such a system might not be available unless the loads are evenly distributed among the three-server. Another proposed approach was the stochastic simulation model [3]. The aim of the authors is to provide an efficient and equitable voting exercise across the designated voting centers. Despite the fact that the stochastic simulation model was used and the server with buffer size determined, the authors failed to show how the load can be re-directed to least loaded server if a server is down or heavily loaded. Load balancing using mobile agents is a potential solution to completely enhance the performance of e-voting system. Mobile agent based approach load balancing in heterogeneous web server system has also been proposed in [9]. The authors used server management agent (SMA), load information agent (LIA) and job delivering Agent (JDA). Load information agent gathered information about a particular load before the information is communicated with the server module agent, where the permission is granted to the job dispatching/ delivering agents to allocate/distribute the load to the recipient servers. This load arrives at a delayed time, and very likely, the load state of the recipient server may have considerably changed from status that was known to server module agent at the time of transferring the load. Another approach similar to [9] was Dynamic Load Balancing Algorithm for Cloud Computing using Mobile Agents proposed by [4].

3.0 **Material and Method**

In this work, we designed a conceptual architecture and flowchart for load balancing in dynamic e-voting system using mobile agents as it is shown in figures 1 and 2.

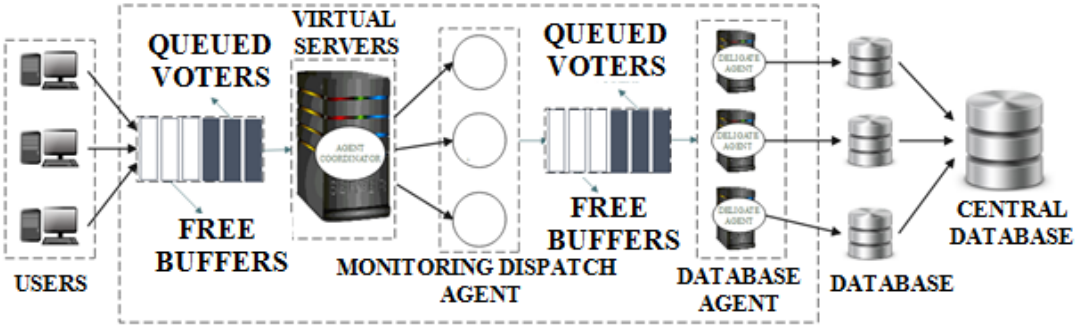

Figure 1: Architecture of load balancing scheme for dynamic e-voting system using mobile

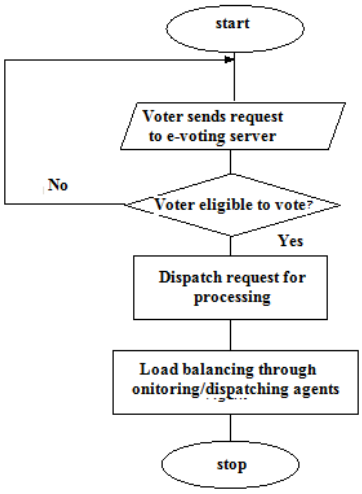

Figure 2: Flowchart for load balancing process in Dynamic voting system

3.1 **Voter's Queue Discipline and Service mechanism**

The Poisson distribution is used to model the arrival rate of the voters, and all voters in the system served on a First-Come-First-Serve basis with no consideration for priority. The equation 1 shows the voter arrival pattern to the e-voting system which has a Poisson arrival pattern as it was described in [15].

The λ_i is a constant $\forall i = 1, 2, ..., n$

Therefore, $\lambda = \lambda_1 + \lambda_2 + ... + \lambda_n$ (2)

Suppose the system is in state E_n, the probability of a vote occurring in a small time interval Δt is considered as $\lambda_n \Delta t + D(\Delta t)$ and that a voter has cast his vote is considered as $\mu_n \Delta t + D(\Delta t), \mathrm{n} \geq 1.$ The system has been in E_{n-1} and had a vote or in state E_{n+1} and had a vote cast done.

Thus, $P_n = \frac{\lambda_{n-1}, \lambda_{n-2}, ..., \lambda_0}{\mu_n, \mu_{n-1}, ..., \mu_1} = \prod_{i=1}^{n-1} \frac{\lambda_i}{\mu_{i+1}}, n \geq 1$ (3)

By mathematical induction

$$P_{n+1} = \prod_{i=1}^{n-1} \frac{\lambda_i}{\mu_{i+1}} p_0 \quad (4)$$

Thus, $P_n = (\frac{1}{n!} \rho^n) \mathrm{e}^{-\rho}, \forall \mathrm{n} \geq 1$ (5)

Here, voting rate increase with increase in queue length. Hence, this is known as the queuing problem with infinite number of channels $(M / M \infty) : (\infty / FIFO).$

3.2 **Optimal Server Size**

In this study, multi-server queue system with infinite buffer is used and all arriving voters wait until they are served. Fig. 3 shows the multi-server system for dynamic voting system where all arriving voters waited until they voted.

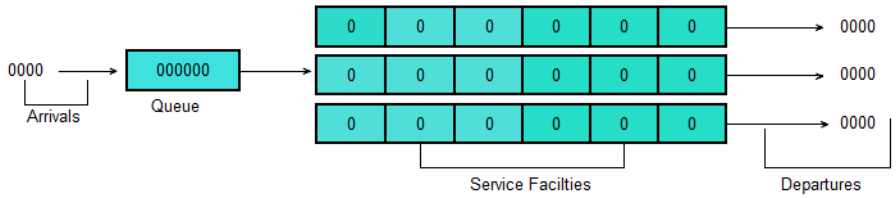

Figure 3: Multi-server system for dynamic e-voting system

In a multi-server environment of $S_i, i = 1, 2, ..., n,$ the optimal number of servers is determined for efficiency of e-voting system. The work of [16] is adapted in this study to determine optimal number of server; this is done when the system is at the steady-state. A load balancer class is denoted by A with arrival rate λ and service rate μ_1 for each queue and a server class is also denoted by S with arrival rate $P_{cr}\lambda$ and service rate μ_2 for each queue. The probability for client requests to require this server class's services is P_{cr}. Since the optimal number of servers in a server class is denoted by S and service rate of server must not be greater than the load balancer, then

$$S\mu_2 \leq A\mu_1 \quad S\mu_2 \leq A\mu_1 \quad (6)$$

Let us assume that $S = Q * M$ (7)

where Q is the server processing speed and M is the number of servers in a server class. Then, the range of M becomes

$$\frac{P_{cr}\lambda}{Q_{\mu_2}} < M \leq \frac{A\mu_1}{Q\mu_2} \quad (8)$$

If equation (8) is satisfied, this indicates that the number of servers M is within the range of the optimal number of servers otherwise the number of servers need adjustment.

3.3 **Buffer**

Buffer used in this work monitors the arrival of voters into the queue and control the order in which the voters are serviced. The order in which voters are serviced is based on First-Come-First-Serve, this controlled the traffic and system response time is improved. The fig. 4 depicts the queue-buffer used and the algorithm 1 adapted from [17]. The algorithm shows the circumstances for allowing the voter into a queue and which voter should be allowed into the voting system

Figure 4: Buffer for dropped control scheme

Algorithm1: A Modified Best Fit Drop Control Scheme

Step 1: Check whether the remaining cache space can accommodate the voter (v)

Step 2: If the remaining space is more than or equal to the number of voter (v)
Allow voter (v) into the queue
Otherwise
Drop some voter (v) from the cache to free enough storage space.
End If

3.4 **Mobile Agent for load balancing and distribution (Agent Execution)**

The agents used in this work have a reactive architecture and the content of agents depends on their responsibilities as it is depicted in figure 5 shows the agents interaction in the dynamic e-voting system.

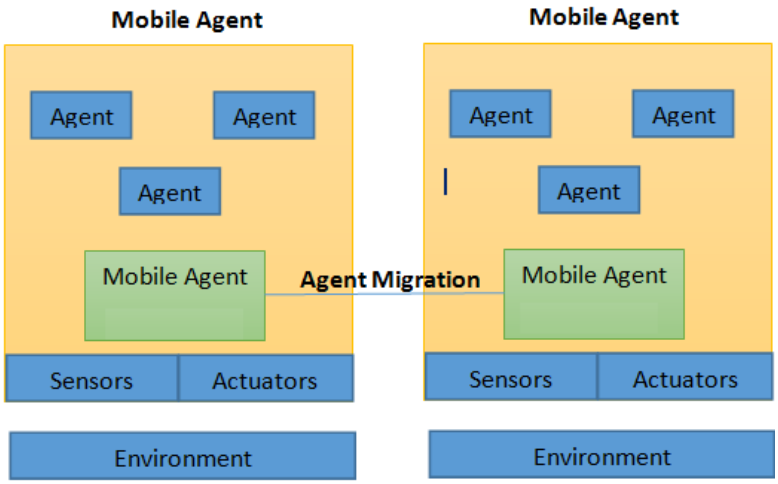

Figure 5: Reactive Architecture for Mobile Agent Structure [18]

3.5. **Agent Responsibilities**

Agent coordinator (AC): This is a manager agent that is responsible for dispatching subordinate agents and coordinates their respective activities in an e-voting network. (AC) manages the voter's request and also coordinates the orderliness of mobile agents' environment. AC uses selection policy based on first-come-first-serve to select voter's message to be transferred for processing and determines the mobile agent to accomplish the task. The processed request will also get back to the voter through the agent coordinator (AC) at the virtual server.

Monitoring/dispatching agent (MDA): This is an actualized mobile agent that is responsible for monitoring to collect information about the server's status by computing composite index of variation and dispatching the load. If a server is heavily loaded the incoming load is redirected to the least loaded server. Once this agent is created (MDA) at the virtual server, it gets all necessary tools to accomplish its task, to reduce delays that may occur during communication/transfer of task between virtual server and other local servers. The composite index of variation (CV) allows the consideration of server processing time and server's capacity to process the task allocated.

Delegate Agent (DA): This is a server's voter agent that resides on local server, to receive incoming message/request (vote) that requires processing from AC through MDA. It also ensures that voter's messages are service on First-Come-First-Serve basis by the server.

3.6 **Transparency distribution and balancing of load (processor capability and load's requirement during run-time)**

The dynamic load balancing approach is used, to distributes and balances the loads during the execution run-time based on the capability of server's processor which can be measured through job wait time and job response time. The MDA handled the load balancing with Kilbridge and Wester's algorithm depicted in fig. 6 to evenly distribute the loads among the servers during the run-time. The agents used algorithm in fig. 7 to ensure that all incoming loads are redirected to the least loaded server if a server is heavily loaded.

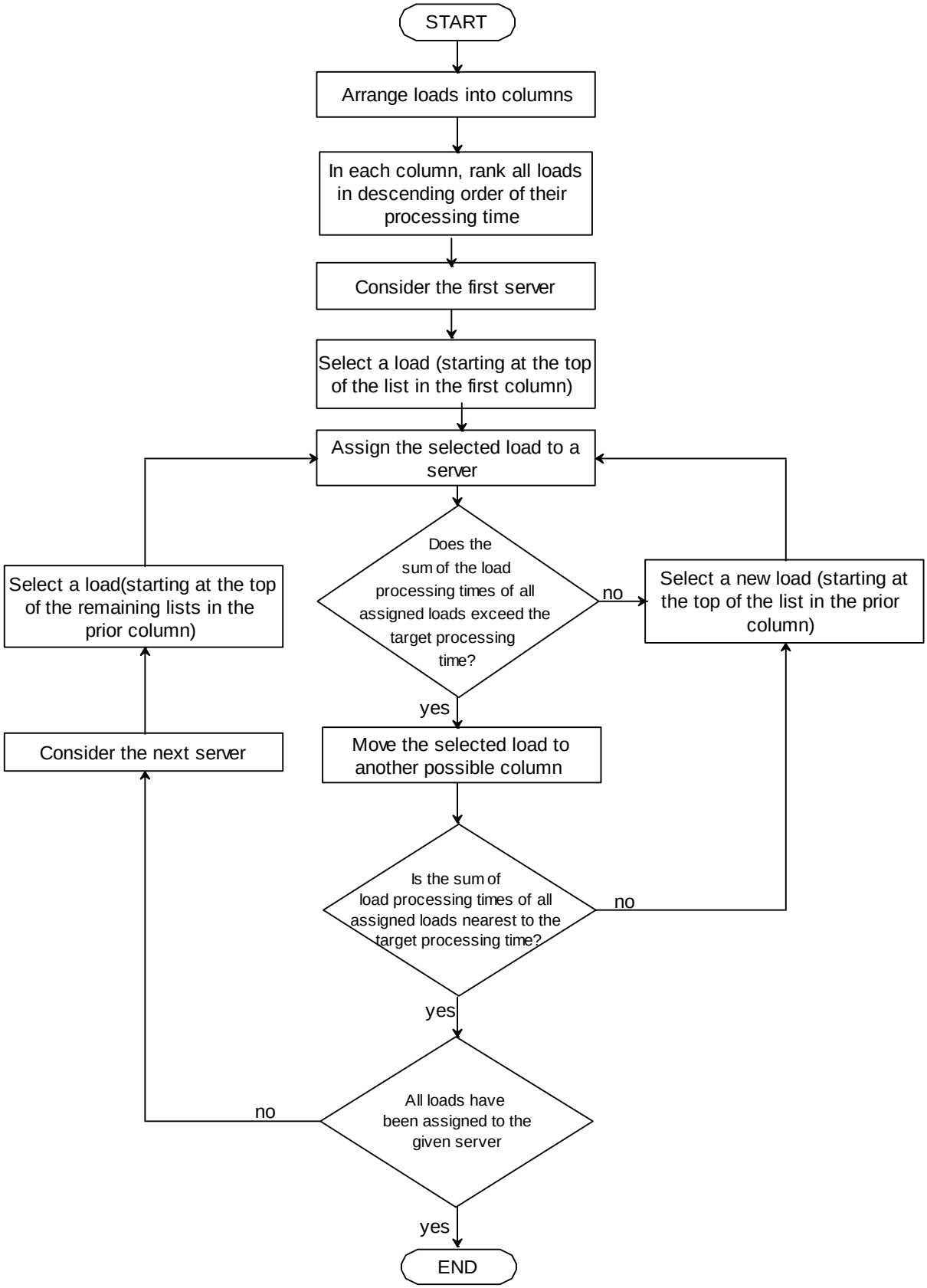

Figure 6: Kilbridge and Wester's heuristic algorithm for Load Balancing

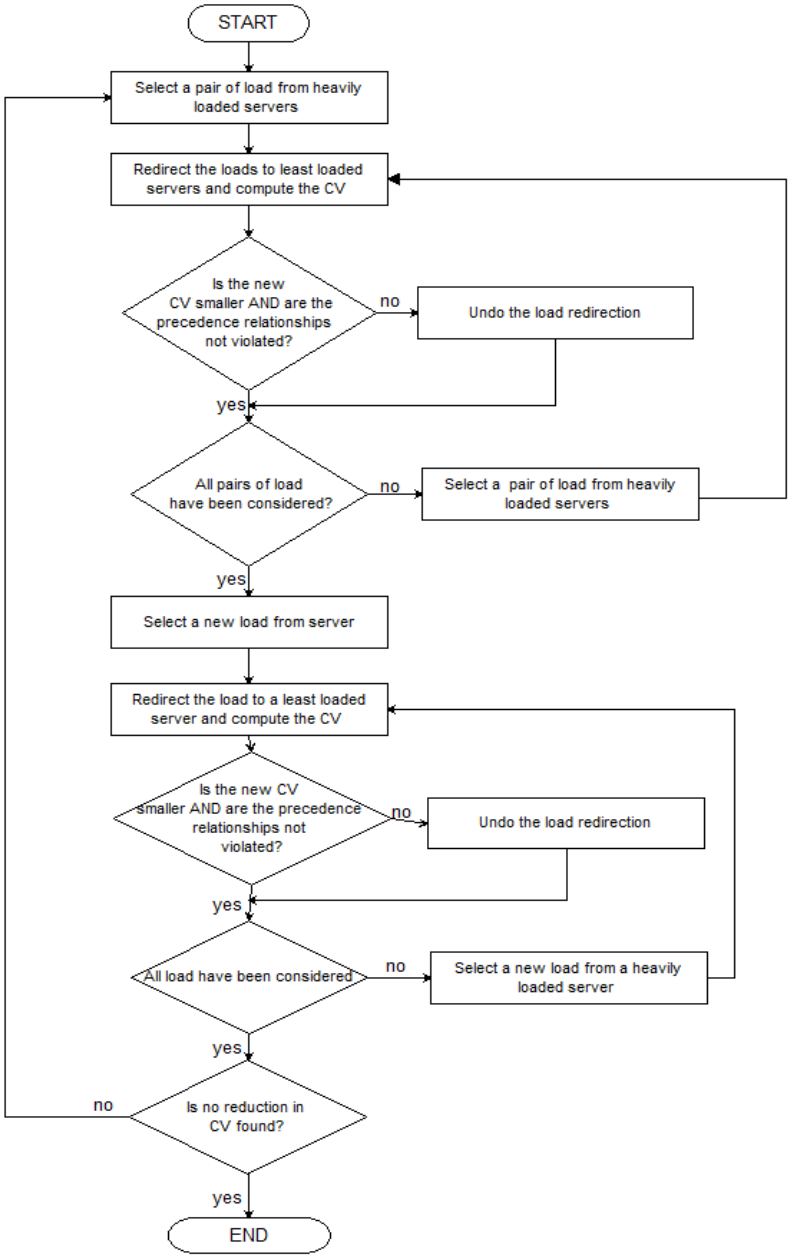

Figure 7: Algorithm for redirecting a new load to least loaded server (Adapted from [19].

3.6 **Hungarian assignment method for load transferring and balancing scheme**

Agent technology used in this work, improves the efficiency of dynamic e-voting system by avoiding delays during communication and transferring of voters' messages (load) among the servers. Once the monitoring/dispatching agent is created at virtual server, it is equipped with all necessary policy needed to distribute and balance the loads so as to minimize the cost and time taken to travel from server (i) to server (j). The task of MDA is similar to that of the travelling salesman problem which is usually represented with equations (9) and equation (10), and may be solved as assignment problem [5].

$$\sum_{i=1}^{M-1} x_{ij} = 1,\ j = 1,2,...,n, i \neq j \qquad (9)$$

Equation (9) indicates that MDA can only make a trip for assigning a load to server (j), while equation (10) shows that MDA has to visit all local servers in the system

$$\sum_{j=1}^{M} x_{ij} = 1, \mathrm{i} = 1,2,...,n-1, i \neq j \qquad (10)$$

Subject to transferring task given to virtual server (i) and processing task given to local/subordinate server (j), the next equations (11) and (12) are used to model the availability of MDA at virtual server (i) and activity requirement of server (j) to process the voters' messages (loads).

$$\sum_{j=1}^{M} x_{ij} = 1, \text{ for all server } i \qquad (11)$$

$$\sum_{j=1}^{M} x_{ij} = 1, \text{ for all server j} \qquad (12)$$

The objective function is then

$$\text{Minimize Z} = \sum_{i=1}^{M-1} \sum_{j=1}^{M} d_{ij} x_{ij} \qquad (13)$$

where each d_{ij} is the distance between virtual server i and local server j.

Since $x_{ij} = 1 \; or \; 0$ for all i and j, then for an agent to collect load from server (i) and assigns it to server (j), implies that the condition that $x_{ij} = 1 \; or \; 0$ is automatically satisfied. The delay is minimized with monitoring/dispatching (MDA) and a load balancing agent that performs the function of information agent and job dispatching agent.

3.7 **Implementation**

The Java Agent Development (JADE) framework was used for implementing the mobile agent's task. The JADE graphical interface (GUI) is provided by a JADE system agent called the Remote Monitoring Agent (RMA) and it allows a platform administrator to manipulate and monitor the running platform. Figures.8 and 9 show the selected agents to process the votes and balance the loads among the servers. These figures also show the simulation result.

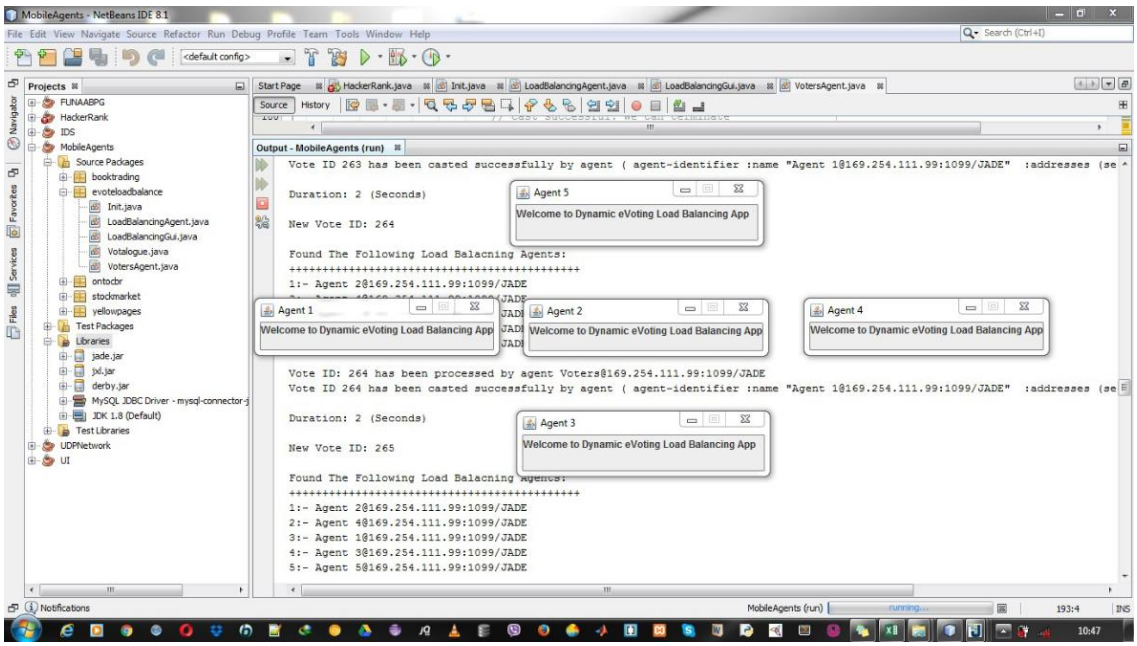

Figure 8: Load Balancing Agents within JADE

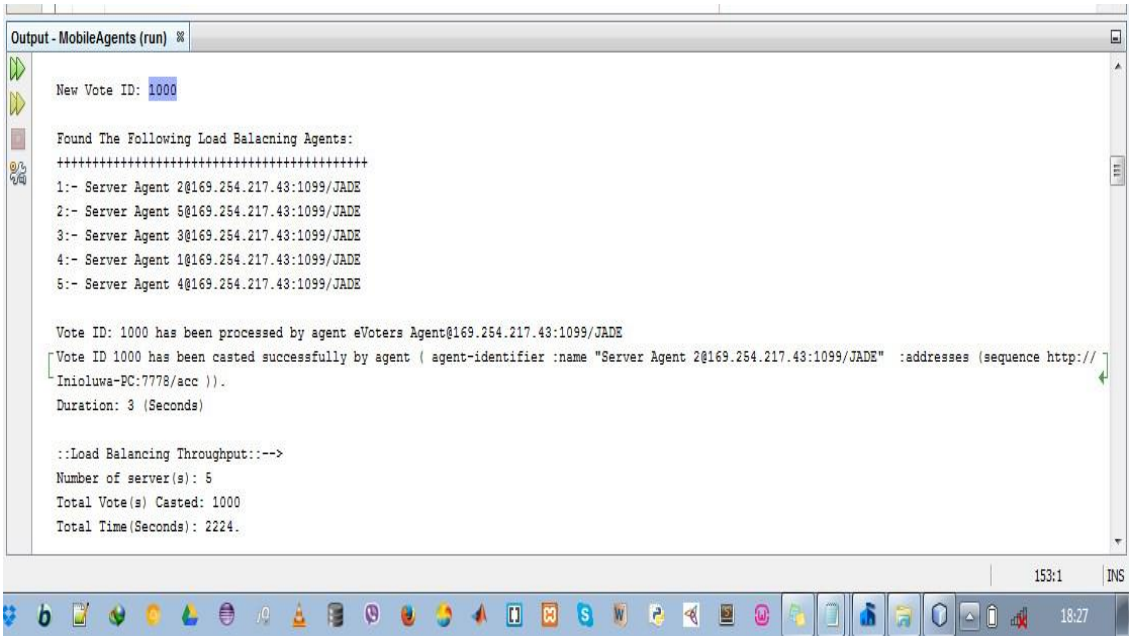

Figure 9: Log of Activities of Load Balancing Agents

3.8 **Result and Discussion**

From the results of our sample runs generated from the load balancing implementation and tabulated in table 1, the maximum response time is 20 seconds while the minimum response time is 2 seconds while the voter's waiting time corresponds to the length of time that he voter stays with the system. The result shows high level of balancing that did not entail queuing. The charts in figures 11 and 12 show system response time, voter's waiting time, queue length and system throughput.

Table 1: Results of Load balancing scheme

Queue Length	Waiting Time (seconds)	Response Time (seconds)	Voter ID	Agent
0	2	2	250	5
0	2	2	189	4
0	3	3	3	4
0	4	4	9	3
0	4	4	11	3
0	5	5	8	3
0	5	5	237	3
0	8	8	5	1
0	9	9	10	1
0	20	20	364	1

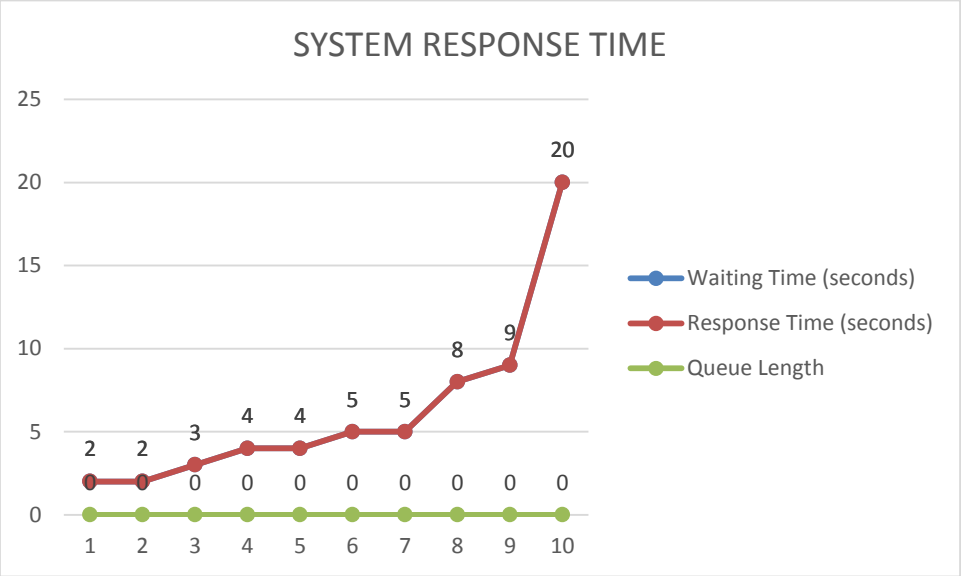

Figure 11: Chart showing the load balancing Results

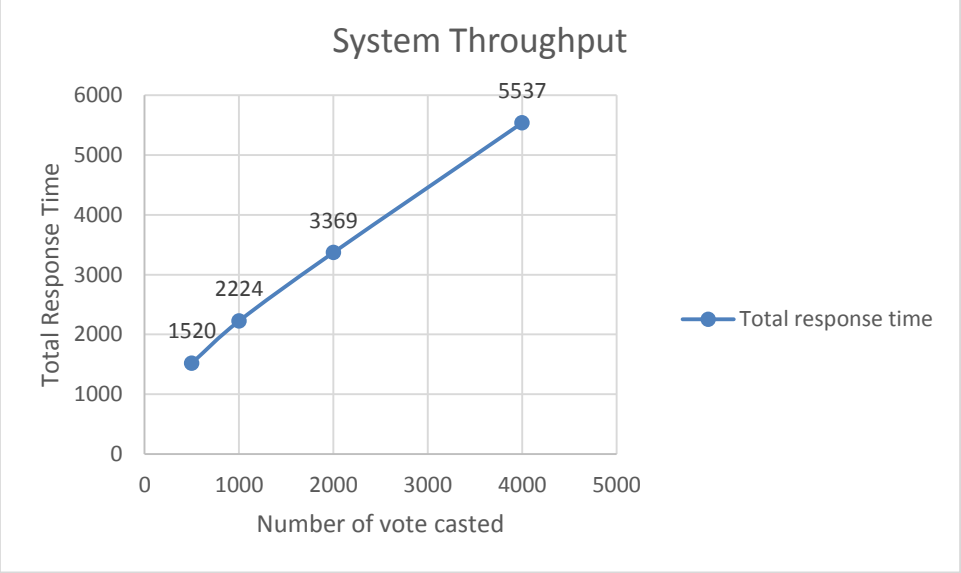

Figure 12: Chart showing the system throughput

4. **Conclusion**

We concluded that load balancing for dynamic e-voting system with Agent Technology improved the performance of dynamic e-voting system. The techniques improved system response time and reduced the voter's waiting time. The load balancing techniques using mobile agents guarantees system availability for large-scale election.

5. **References**

[1] Mohammed, I. A. and Mohammed, A. (2013). Internet voting: Security and Performance Issues. Faculty of Computing and Information Technology, Egyptian *Computer Science Journal ECS. 37(4), pp. 92-106*

[2] Mohammed, M., Mohammed, K. & Omar, A. (2009). Modelling and Simulation of a Robust E-voting System, *Communications of the IBIMA, Vol.* 8, *pp. 198-206.*

[3] Ugbebor, O., O. and Nwoye C., (2012). Modelling and Analysis of the Queue Dynamic in The Nigeria Voting System. *The Open Operation Research Journal, vol. 6, pp. 9-22.*

[4] Nandita, G. (2013). Dynamic load balancing algorithm for cloud computing using mobile agent, *Information Technology Department, ABES Engineering College, Ghaziabad, India*, 3(12), pp. 477-479.

[5] Sharma, J. K. (2009). *Operations Research Theory and Applications, 4th Edition, Macmilla Publishers India Ltd.* pp. 583.

[6] Leinberger, W., Karypis, G. & Kumar, V. (2000). Load Balancing Across Near-Homogeneous Multi-Resource, Servers, *Proceedings of the Heterogeneous Computing Workshop (HCW), IEEE, pp. 60-71.*

[7] Ezumalai, R., Aghila, G. & Rajalakshmi, R. (2010). Design and Architecture for Efficient Load Balancing with Security Using Mobile Agents. *IACSIT International Journal of Engineering and Technology, 2(1), pp. 57-58.*

[8] Mohammed, A.M.I., (2010). "Cluster of heterogeneous computers: Using mobile agents for improving load balance", *International Journal of Science and Technology Education Research Vol. 1(7), pp. 143 – 146.*

[9] Jyoti, V. & Anant, K.J. (2014). Mobile Agent Based Approach to Load Balancing in a Heterogeneous Web Server System, *IOSR Journal of Engineering (IOSRJEN), 4(6), pp. 44-47.*

[10] Sameena, N., Afshar, A. & Ranjit, B. (2012). *Load Balancing Algorithms for Peer to Peer and Client Server Distributed Environments*, International Journal of Computer Applications (0975 - 888), Vol. 47, No.8, pp. 17-19.

[11] Majeed, M.H., Sagar, D., Chaouki, T. A., Douglas, B. J. and John, C., (2003). Dynamic Time Delay Models for Load Balancing Part II: A Stochastic Analysis of the Effect of Delay Uncertainty. *CNRS-NSF Workshop: Advances in Control of Time-Delay Systems, Paris, France, pp. 1-16.*

[12] James, I. N. (2014). E-voting System: A Simulation Case Study of Kenya, *M.Sc. Thesis Submitted to the School of Computing and Informatics, University of Nairobi .Kenya.*

[13] Aneta, Z. and K.Zbigniew, K., (2006). An Efficient Agent e-Voting System with Distributed Trust, *VODCA 2006, Preliminary Version. pp.* 86-88.

[14] Anichebe, N. A. (2013). Queuing Model as a Technique of Queue Solution in Nigeria Banking Industry, *International Institute for Science, Technology and Education (IISTE), 3(8), pp.188-195.*

[15] János, S. (2012). *Basic Queueing Theory*, CA: University of Debrecen, Faculty of Informatics, *pp.16 & 35*

[16] Jin, H.S. and Myoung, H.K., (2004). An Analysis of the Optimal Number of Servers in Distributed Client/Server Environments, *Elsevier Science, 36, 297– 312.*

[17] Yechang Fang, K.Y. and Deng Pan, Z.S., (2010). Buffer Management Algorithm Design and Implementation Based on Network Processors. *(IJCSIS) International Journal of Computer Science and Information Security, 8(1), pp. 1-8.*

[18] Tim J. M. (2008). Artificial intelligence: A System Approach, *Infinity Science Press Llc*, Hingham, Massachusetts, New Delhi, pp. 371.

[19] Chorknew, J., Suebsak N. and Sanchoy K.D., (2013). Heuristic Procedure for the Assembly Line Balancing Problem with Postural Load Smoothness, *International Journal of Occupational Safety and Engineering, (JOSE),19(4), pp. 531-541.*