Volume 7, Issue 2, February 2017 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Optimization of Distributed System through Load Balancing and Task Scheduling

[1]Abhijit A. Rajguru, [2]Dr. Sulabha S. Apte

[1]Research Scholar at Walchand Institute of Technology, Solapur, Maharashtra, India

[2]Walchand Institute of Technology, Solapur, Maharashtra, India

DOI: 10.23956/ijarcsse/V7I2/0117

Abstract— Load balancing problem in distributed network system are the challenging research area in computer science and engineering. In distributed system network, task scheduling and load balancing mechanism has lots of issues as there is no centralized system to assign the work-load among multiple processing nodes. In this research paper, a fuzzy based load balancing and task scheduling technique is proposed to enhance the performance of distributed network system. Firstly, clusters are created and coordinator node is elected which has bigger memory size and high CPU speed. Tasks are divided into flexible and non-flexible using task prioritizing strategy. Non-Flexible tasks have more priority than flexible tasks. For non-flexible tasks, required information of computing nodes such as CPU speed, work-load and distance from cluster coordinator are passed through the fuzzy system. As an output, node state is obtained. Non-flexible tasks are assigned to processing node according to node states. A simulation result shows the efficiency of our technique.

***Index Terms— Distributed Computing, Fuzzy Logic, Load Balancing, Task* Scheduling.**

I. INTRODUCTION

1.1 Distributed Systems

A distributed system network consists of set of communication and computing resources, shared by active users [1]. Resource sharing, reliability, scalability, economy etc. are the features of distributed system. [2, 3]. Due to time accuracy distributed system network is mainly useful in telecommunication and computer network. Distributed system has many applications like aircraft and industrial control system, etc. [4]

1.2 Load balancing

In parallel and distributed network systems, many tasks or jobs are processed parallelly by multiple processors. The process of equalizing work-loads among the processing node is known as load balancing technique. Load balancing attains good throughput. [5, 6]

Load balancing can be achieved by statically or dynamically. Based on this, it is divided into static load balancing and dynamic load balancing. [7]

a. Static Load- balancing

Static load balancing uses prior knowledge of a system and application to balance the work-load.

b. Dynamic load-balancing

In dynamic load balancing, work-loads are assigned to computing node dynamically to balance the load [8].

As demand increases, problem of load balancing becomes significant. The main objective of load balancing technique is to enhance the distributed network system performance [9].

1.3 Scheduling in distributed systems

Task scheduling technique is also used to enhance the performance and throughput of distributed network system. [10]. The process of distributing tasks/jobs between computing nodes is known as scheduling. Task scheduling technique is divided into local scheduling and global scheduling.[5]

a. Local scheduling

In local scheduling technique processes are allocated to the time slices of the processor.

b. Global scheduling

In global scheduling technique scheduling of processes are done by master node. Global scheduling is classified into two types as static global scheduling and dynamic global scheduling [5].

1.4 Issues of load balancing and task scheduling.

The load balancing and task scheduling technique has more issues, some are described below:

1) Load balancing and task scheduling in the distributed network is difficult [5].
2) Load balancing and task scheduling technique should support characteristics like scalable and low overhead [7].
3) Managing both task scheduling and load balancing at a time is difficult task [10, 12].

II. RELATED WORK

Zalinda, [7] uses fuzzy inference system to decide robots based on their reliability. The fuzzy system balances and monitors the load presented to the robots. Fuzzy logic was used to model the reliability and efficiency of the robots under the various conditions.

Shaout [10] used fuzzy load balancing algorithm for a small database driven distributed network. The presented algorithm reduced wait times, total times, and average run time. The fuzzy system worked especially well for high priority traffic (traffic that demanded a quick response time).

Dierkes [11] realised the uncertainty that was present in all load balancing algorithms and developed a fuzzy set to estimate the impact of the algorithm's estimation of the current cluster state. In another paper [12], Dierkes based load balancing decision making on a set of possible actions, a set of goals with priorities, and an effect matrix. The effect matrix showed the effect of the actions on the goals (whether the action either distracts or supports a goal). The application of fuzzy sets taken from the matrix significantly improved system performance, with a bias towards achieving the goals given weight in the effect matrix.

Park [13] developed algorithm called "the imprecision in state information and makes scheduling decisions based on fuzzy logic". Fuzzy logic states were used to model uncertainty, mainly in the system load and message propagation measurements. Each scheduler had two functions, threshold estimation and decision making. The two thresholds indicated the amount of variation between its current load and the servers around it; and the amount of variation between its current load and its external load (what it was working on and what it was yet to work on). The algorithm achieved an improved mean response time over conventional algorithms as well as a low constant message overhead cost.

In paper [14], author introduced a new protocol for load balancing using fuzzy logic control,multicast, and active network concepts. This new protocol is based on a logical hierarchical structure that locates nodes for load transfer dynamically. We use multicast in order to save the bandwidth of the network. Nodes in the computer networks can adjust

their membership functions according to their load status and traffic congestion status. An interesting feature of the protocol is that it first tries to find nodes at close distance for the load transfer over remote nodes. The protocol is also stable and smooth because no fixed threshold levels are used and there is a smooth transition from lightly or heavily loaded status to moderately status. Currently we are simulating this protocol and comparing its performance with other protocols in a computer networks.

III. FUZZY BASED LOAD BALANCING & TASK SCHEDULING TECHNIQUE

3.1 Overview

In this paper, we propose a fuzzy based load balancing and task scheduling technique for distributed network systems. The proposed network contains a clients (C's) and servers (S's) nodes. From these nodes cluster coordinator node is elected which has more memory size and CPU speed. The elected coordinator node (CN) is responsible for coordinating and scheduling the processes or tasks. The processes or tasks are prioritized using Boolean value. 1 denotes Flexible tasks whereas 0 denotes non-flexible tasks. When client receive any new job, first its information is transferred to cluster coordinator node (CN). The coordinator node (CN) looks for flexible field. For non-flexible job/tasks, it collects required information such as work-load of nodes, CPU speed and distance from CN. This information is passed through fuzzy system. As an output of fuzzy system, server nodes are classified into three states as, schedulable (SC), likely to be schedulable (LS) and not schedulable (NS). Non-flexible jobs are assigned to server nodes in schedulable state and when buffer requirement exceeds the buffer availability of SC state nodes, and then they are assigned to server nodes in LS state. The flexible tasks are assigned to server nodes in likely to be scheduled (LS) state.

3.2 Computation of Metrics

3.2.1 Available Buffer Estimation

The buffer availability of node 'n' at time't+1' is calculated as:

$$AvaB_{ni}^{t+1} = \left(\frac{\sum_{n=0}^{I-1}((I-n)\cdot AvaB_{ni}^{t-n}}{\sum_{n=0}^{I-1} n} \right) \bullet p \qquad (1)$$

Where, I denote iteration number, n stands for nodes in the distributed network system, $AvaB_{ni}^{t-n}$ is the available memory computed before at t-x and p is memory estimating probability, usually set to 0.95.

3.2.2 Work load calculation

Work load (W_L) at a processing node is calculated as follows:

$$W_{Li} = N_T \times S_T \qquad (2)$$

Where, N_T is the number of tasks/jobs to be computed and S_T size of tasks/jobs.

3.3 Cluster Coordinator Node Selection

In each cluster, a cluster coordinator node (CN) is selected by considering larger memory size and high CPU speed. The process of selection is given below:

(i) Clients (C's) and server (S's) nodes forms a clusters.

(ii) Every node estimated available memory value ($AvaB_{ni}^{t-n}$) using equation (1)

(iii) Every computing node sends Nw Info (Network Information) message in the cluster. Nw Info message includes $AvaB_{ni}^{t-n}$ value and CPU speed.

$$\text{Computing Node} \xrightarrow{NwInfo} \text{Cluster}$$

(iv) Every computing node receives the broadcasted value and finally a node with more memory size ($AvaB_{ni}^{t-n}$) and high CPU speed is selected as cluster coordinator node (CN).

(v) The elected coordinator node (CN) broadcasts this selection information to all cluster member.

$$\text{CN} \xrightarrow{CHselecion} \text{Nodes in the Cluster}$$

3.4 Task Prioritization Strategy

Generally, every task consists of task size, unique task_id, and deadline fields. In addition to these fields, an additional Boolean field is added to identify the priority of tasks/jobs. The Boolean value 1 represents flexible and 0 represents non-flexible tasks/jobs. Here we set priority to non-flexible task over flexible because the deadline of non flexible task should not be missed. For flexible task, we can expand execution time if its deadline is missed. The header of task is shown in table-1.

Table-1 Fields in Task Header

Task_ id	Task Size	Deadline	Flexible/ Non-flexible

3.5 Fuzzy Based Task Scheduling Technique

When client node receives any new task/job to be processed, it transfer task/job information details to the coordinator node (CN). The coordinator node (CN) first looks for flexible field of task/job. If the Boolean value is zero then coordinator node (CN) collects information of server nodes like work-load, CPU speed and distance from the CN. This information's of every server node (S) are passed through the fuzzy system to obtain states of server nodes like as schedulable (SC), likely to be schedulable (LS) and not schedulable (NS).

3.5.1 Fuzzification

In fuzzification phase, the input values of nodes like work-load, CPU speed and distance from coordinator node (CN) are fuzzified by using membership functions (MF). The input values of work-load, CPU speed and distance for membership functions are given in figure-1, 2 and 3 respectively. Figure-4 represents the membership function of output value.

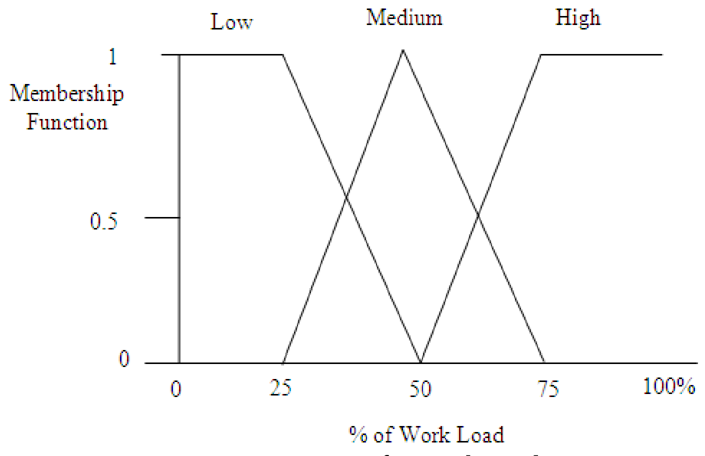

Figure 1: Fuzzy Set for Work Load

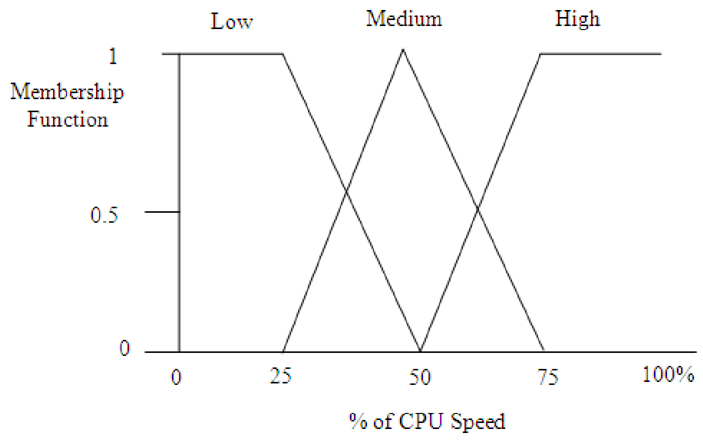

Figure 2: Fuzzy Set for CPU Speed

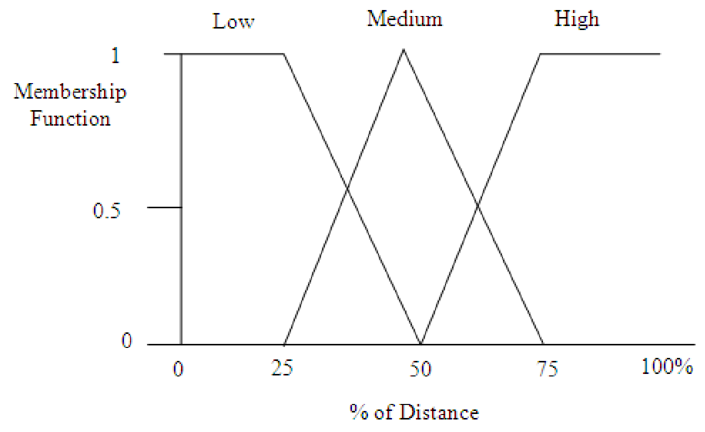

Figure 3: Fuzzy Set for Distance

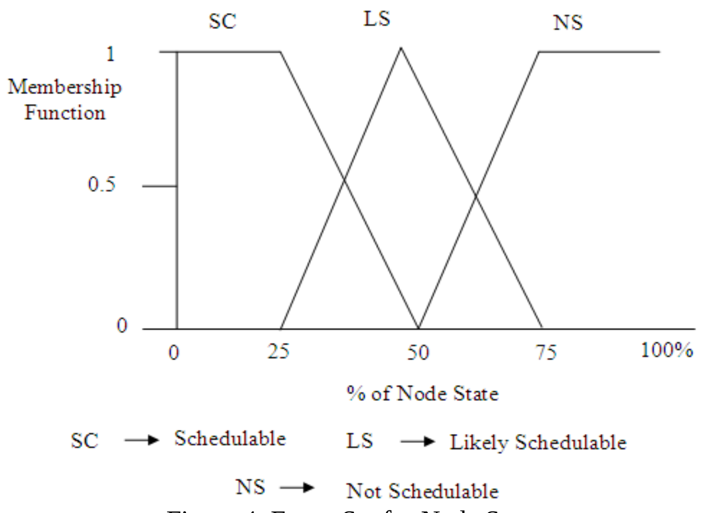

Figure 4: Fuzzy Set for Node State

3.5.2 Fuzzy Inference Engine.

In our fuzzy system, fuzzy inference system is built by considering the following 13 rules as shown in table-2.

Table-2 Fuzzy Rule Set

Rule	**Work Load**	**CPU Speed**	**Distance**	**Node State**
1	Low	Low	Low	LS
2	Low	High	Low	SC
3	Low	High	Medium	LS
4	Low	Low	High	NS
5	Low	High	High	LS
6	Medium	High	Low	SC
7	Medium	High	Medium	LS
8	Medium	Low	Medium	NS
9	Medium	Low	High	NS
10	High	Medium	Low	NS
11	High	Medium	Medium	NS
12	High	High	Low	LS
13	High	Low	High	NS

In table-2, SC-denotes schedulable state, NS- denotes not schedulable state and LS denotes the state of likely to be scheduled. Among 13 rules, we explain the following two rules,

```
(1)  If (Work-Load = Low && CPU_Speed = High && Distance = Low) Then
        "Node State is Schedulable"
     End if
```

(2) If (Work-Load = High && CPU_Speed = Low && Distance = High) Then
"Node State is Not Schedulable"
End if

3.5.3 Defuzzification

For defuzzification process weighted mean method is used.

$$O^* = \sum \eta_o(\bar{O}) / \sum \eta_O(\bar{O}) \qquad (3)$$

where, O^* is the derived output value, $\eta_O(\bar{O})$ represents the strength of output membership function and $\bar{O}$ is the centroid of membership function.

3.5.4 Task Allocation

While assigning tasks/jobs, the CN checks task length of non-flexible tasks and compare the memory length of server nodes in schedulable state (SC). Then, coordinator node (CN) assigns non-flexible tasks to server nodes in schedulable state (SC). If number of server nodes in SC (schedulable state) is less than task-length of non-flexible tasks, then the CN looks for nodes in likely to be scheduled (LS) state. Once, the assigning of non-flexible tasks are gets over, the CN goes for flexible tasks. The task allocation strategy is shown in algorithm-1.

Algorithm-1

```
1. Let S and C be the set of server nodes and the set of client nodes respectively.
2. Consider nTi be the set of non-flexible tasks and fTi be the set of flexible tasks.
3. Client node receives a new task.
4. Client transmits task details to coordinator CN
5. CN looks for flexible field
6. If (flexible field is= 1) Then
        6.1 The task is flexible
        6.2 Else if (flexible field = 0) Then
        6.3 The task is non-flexible
7. End if
8. For (non- flexible tasks)
   8.1 CN gathers work-load, CPU speed and distance
       information's of server nodes from cluster.
   8.2 CN forwards this information's to fuzzy system
   8.3 Nodes are divided into three states as SC, LS
       and NS
   8.4 CN compares nTi task length with buffer
       availability of Server nodes
      8.4.1 If (task length (nTi) < S (buffer availability))   Then
           8.4.1.1. Allocates nTi among Server nodes in SC state
      8.4.2 Else if (task length (nTi) > S (buffer availability))
            Then
           8.4.1.2 Allocate nTi among Server nodes in LS state
      8.4.3 End if
9. For (flexible tasks)
   9.1 CN allocates tasks among S' nodes in LS state
```

IV. SIMULATION RESULT

4.1. Simulation Model and Parameters.

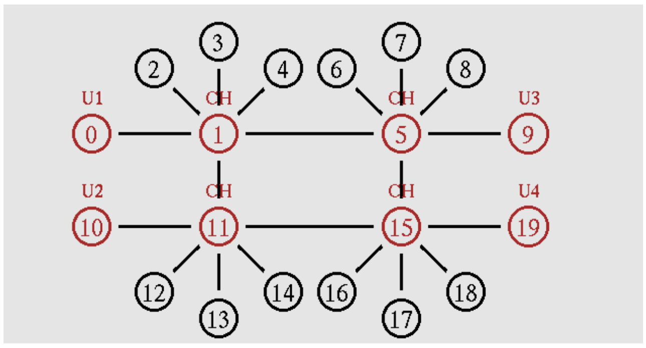

Figure: 5. Simulation Setup

In this section, the performance of our Fuzzy based Load Balancing and Task Scheduling Technique is examined (FLBTS) using NS-2 [17]. The simulation topology is given in Figure 5. We compare our results with QBS(Queue Based Scheduling) with normal scheduling [11]. The various simulation parameters are given in table 3.

Table: 3 Simulation Settings

Mobile Nodes	12
Users	4
Area Size	1000 X 1000
Mac	802.11
Radio Range	250m
Simulation Time	50 sec
Traffic Source	CBR
Packet Size	512
Rate	1,1.5,2,2.5 and 3 Mb

4.2. Performance Metrics

In our experiments, we measure the following metrics.
Scheduling delay, Throughput, Data Loss, Success Ratio

4.3. Results

A. Based on Users

In this experiment, we vary the number of users from 1 from 4 each requesting variable number of tasks with size 1Mb.

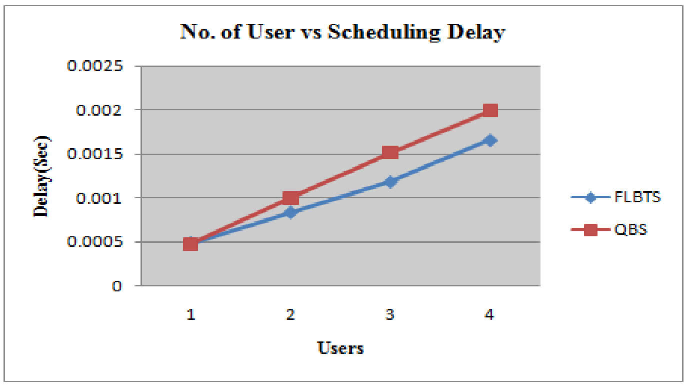

Figure 6: Users Verses Delay

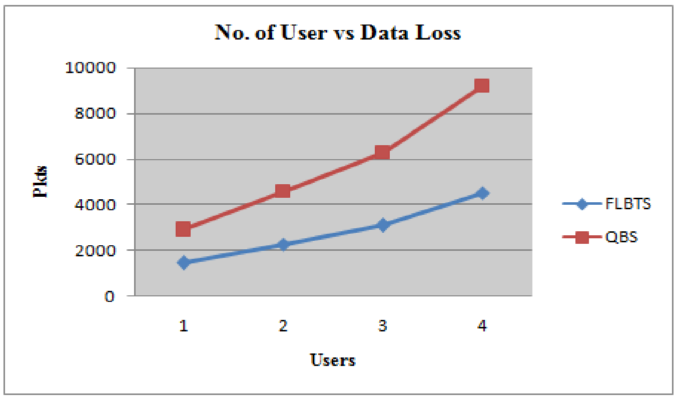

Figure 7: Users Verses Drop

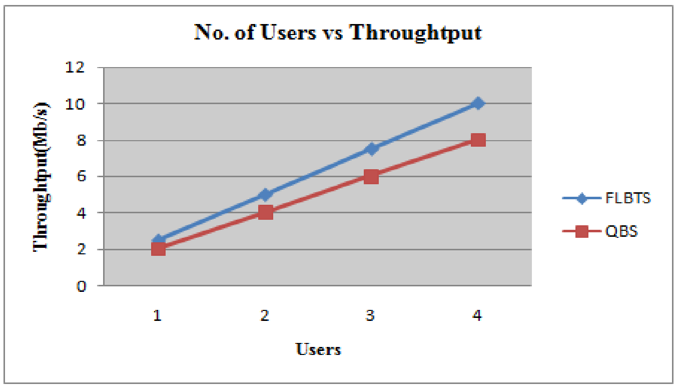

Fig 8: Users Verses Throughput

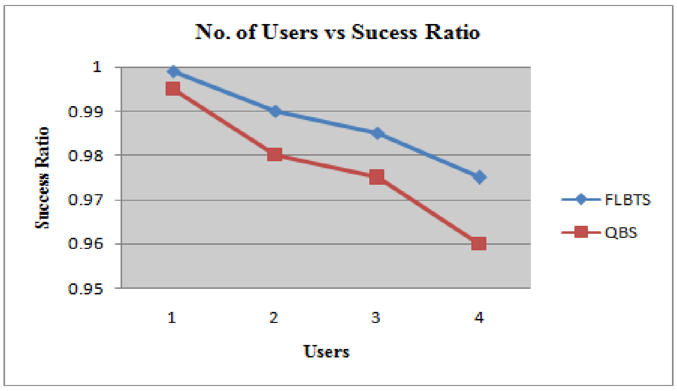

Fig 9: Users Verses Success Ratio

Figure 6 and 7 shows that delay and packet drop of FLBTS is significantly less than the existing QBS technique respectively.

Figure 8 and 9 shows that the throughput and success ratio of FLBTS is higher than the existing QBS method respectively.

V. CONCLUSION

In this paper, we have proposed a fuzzy based load balancing and task scheduling algorithm to enhance the performance for distributed network systems. The proposed technique uses fuzzy logic which has more power in solving load balancing problem. Our technique dynamically manages task scheduling and load balancing at a time. The simulation result shows that efficiency of our technique.

REFERENCES

[1] Daniel Grosu, Anthony T. Chronopoulos and Ming-Ying Leung, “Load Balancing in Distributed Systems: An Approach Using Cooperative Games”, Proceedings of the 16th International Parallel and Distributed Processing Symposium, (IPDPS’02), 2002.

[2] Veeravalli Bharadwaj, Depasish Ghose and Thomas G. Robertazzi, “Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems”, Journal of Cluster computing, Vol-6, Issue-1, 2003.

[3] Krishna Nadiminti, Marcos Dias de Assunção, and Rajkumar Buyya, “Distributed Systems and Recent Innovations: Challenges and Benefits”, Grid Computing and Distributed Systems Laboratory, 2006.

[4] www.wikipedia.org.

[5] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, “Performance Analysis of Load Balancing Algorithms”, World Academy of Science, Engineering and Technology, 2008

[6] Hisao Kameda, El-Zoghdy Said Fathyy and Inhwan Ryuz Jie Lix, “A Performance Comparison of Dynamic vs. Static Load Balancing Policies in a Mainframe { Personal Computer Network Model”, Proceedings of the 39th IEEE Conference on Decision and Control, 2000.

[7] Zalinda, Othman, Khairanum Subari and Norhashimah Morad, Application of Fuzzy Inference Systems and Genetic Algorithms in Integrated Process Planning and Scheduling. International Journal of The Computer, The Internet and Management, Vol. 10, No2, 2002, pp. 81 – 96.

[8] Chow KP and Kwok YK, “On load balancing for distributed multiagent computing”, IEEE Transactions on Parallel and Distributed Systems, Vol- 13, pp- 787-801, 2002.

[9] Jiani Guo and Laxmi Narayan Bhuyan, “Load Balancing in a Cluster-Based Web Server for Multimedia Applications”, IEEE Transactions On Parallel and Distributed Systems, Vol-17, 2006

[10] Shaout A. and McAuliffe P. Job scheduling using fuzzy load balancing in distributed system. Electronics Letters, 1 Oct 1998, 34:1983–1985, 1998. 3. Sacha Dierkes. Extension of Bayesian decision theory and fuzzy logic to decision machines for load balancing.

[11] S. Dierkes. Load Balancing with a Fuzzy Decision Algorithm. Proceedings of the International Panel Conference on Soft and Intelligent Computing, Budapest, Hungary, 7- 10. October 1996.

[12] C. Park, J.G. Kuhl, A fuzzy based distributed load balancing algorithm for Large Distributed Systems. Proceeding of the Second International Symposium on Autonomous Decentralized Systems (ISADS'95) April 25 - 27, 1995 Phoenix, Arizona, USA. IEEE 1995.

[13] Ming-Chang Huang, S. Hossein Hosseini1 and K. Vairavan, Department of Electrical Engineering and Computer Science, “ Load Balancing in Computer Networks”, University ofWisconsin – Milwaukee

[15] Ananda Basu1, Saddek Bensalem, Doron Peled, and Joseph Sifakis,” Priority Scheduling of Distributed Systems Based on Model Checking”, Proceedings of the 21st International Conference on Computer Aided Verification (CAV ‘09), 2009.

[16] Zuo Jing, Chi Xuefen, Lin Guan and Li Hongxia, “Service-aware Multi-constrained Routing Protocol with QoS Guarantee Based on Fuzzy Logic”, IEEE 22nd International Conference on Advanced Information Networking and Applications - Workshops, (AINAW), pp- 762 - 767, 2008.

[17] NetworkSimulator:http:///www.isi.edu/nsnam/ns

ABOUT AUTHOR

Abhijit Rajguru received the B.E and M.Tech degree in Computer Science & Engineering, from Shivaji University, Kolhapur, Maharashtra(INDIA) in 2007 and 2009, respectively. He is research scholar at Walchand Institute of Technology, Solapur (Solapur University, Maharashtra, INDIA).
His research interest includes Distributed System, cloud computing, grid computing, load balancing.

Dr. Mrs. S. S. Apte received Ph.D. degree in Computer Engineering. She having 33 years teaching experience and 02 years industry experience. She is working as professor in CSE department in WIT Solapur. Her research interest includes Computer architecture, distributed system, image processing.