New Fuzzy Techniques for Real-Time Task Scheduling on Multiprocessor Systems

Matthew T. Ogedengbe[1], Moses A. Agana[2]

[1]*Department of Mathematics/Statistics/Computer Science, University of Agriculture Makurdi, Nigeria.*

[2]*Department of Mathematics/Statistics/Computer Science, University of Agriculture Makurdi, Nigeria.*

Abstract - *Real-time computing is rapidly gaining more technological advancement whereby real-time tasks were been scheduled and programmed on computer systems within a time constraint. In this paper, a new fuzzy scheduling algorithms (NFSA) for real-time tasks was proposed which comprises of Arrival time, Computation time, and Deadline as the input scheduling parameters. The proposed NFSA was compared with the Existing Fuzzy Algorithm (EFA) algorithm for performance evaluations. The FEDF algorithm comprises of two inputs scheduling parameters which are deadline and external priority. Tasks were scheduled on multiprocessor at higher system load using fuzzy techniques for both (NFSA and EFA) algorithms. The outputs (runtime priorities) of the simulation were used to schedule tasks in an internal priority (ready) queue for execution on multiprocessor. Results show that the NFSA has a better performance compare to EFA at higher system load. The following performance metrics were considered for the evaluation; minimum response time, turnaround time and number of deadline missed.*

Keywords — *Fuzzy Logic, membership function, Real-time systems, task, multiprocessor scheduling.*

I. Introduction

In modern world of computing, real-time systems play a vital role without which human daily activities can not be carried out conveniently or successfully [1]. We make use of various household real-time devices in our daily activities but know little or nothing about them. From mobile to missile, medical imaging systems, industrial control systems, display systems, Space Shuttle avionics system, traffic control, automated factory, military systems and various scientific experiments [2] require real-time communication and computation. In real-time systems scheduling has more critical role than non-real-time systems because in these systems having the right answer too late is as bad as not having it at all [3]. Such a system must react to the requests within a fixed amount of time which is called deadline. Modern embedded computing systems are becoming increasingly complex. Scheduling real-time systems involves allocation of resources and CPU-time to tasks in such a way that certain performance requirements are met.

Categorically, real-time systems can be categorized into two important groups: hard real-time systems and soft real-time systems. In hard real-time systems, all deadlines must absolutely be met or the system will be considered to have failed (system failure might be disastrous), while in soft real-time systems some deadlines maybe at least occasionally missed with only a degradation in performance but not a complete failure (i.e. missing some deadlines is tolerable) [4]. In both cases, when a new task arrives, the scheduler is to schedule it in such a way that guaranties the deadline to be met. These tasks can be classified as periodic or aperiodic. Periodic tasks are type of tasks that occur at regular intervals, and aperiodic tasks occur unpredictably. The length of the time interval between the arrivals of two consecutive requests in a periodic task is called period.

The multiprocessor based scheduling have more computational complexity in practical algorithm which are unknown to most researchers, this open floor for new area of research in operating systems [5]. Multiprocessor scheduling techniques in real-time systems fall into two general categories: partitioning and global scheduling. Under partitioning, each processor schedule tasks independently from a local ready queue. Each task is assigned to a particular processor and is only scheduled and executed on that processor. While in global scheduling all ready tasks are stored in a single queue. The highest priority task is selected to execute whenever the scheduler is invoked. It has been proved that finding a minimal schedule for a set of real-time tasks in multiprocessor system is NP-hard [6]. However, in both cases researchers have made some significant contributions by those results in better multiprocessor scheduling algorithms.

The scope of this paper is limited to the scheduling of periodic task scheduled on multiprocessors in a soft real-time environment with fuzzy parameter constraints. In section 2 related works were reviewed while section 3 discussed the common scheduling algorithms used in real-time systems. Section 4 discussed the fuzzy inference system, the proposed model and proposed algorithm. Section 5 discussed the results and performance evaluation. Finally, section 6 comprises of conclusion and future work.

II. RELATED WORKS

Many Researchers have employed various fuzzy techniques to schedule tasks in the recent years in order to obtain an optimal performance, but this area of scheduling on multiprocessors is still an open problem [6].
Mahdi *et al.*, proposed in [7] a fuzzy scheduling approach to arrange real-time periodic and non-periodic tasks in multiprocessor systems. Their approach successfully balanced task loads on multiprocessor by using fuzzy scheduler. They stated that higher priority tasks have higher running probability. Their results show that the proposed fuzzy scheduler creates feasible schedules for homogeneous and heterogeneous tasks. Shatha *et al.*, [8] presents a paper; *A fuzzy-based CPU scheduling algorithm*. Their work applied fuzzy logic in the design and implementation of a rule-based scheduling algorithm to solve the shortcoming of priority scheduling algorithms. Task priority and execution time was used as the input parameter for the fuzzy engine while shortest job first only considered the execution time. Their results demonstrate that the average waiting time and the average turnaround time in their proposed algorithm are better than that obtained using priority scheduling, and closed to that obtained from shortest-job-first (SJF) scheduling. Sheo *et al.*, [6] propose a fuzzy approach to multiprocessor real-time scheduling. Their fuzzy model consists of two input parameters, priority and deadline. Their algorithms were examined based on load factor and other performance metric (such as deadline miss, CPU utilization, response time and turnaround time) for system performance evaluation. Their results show that using deadline as a fuzzy parameter in multiprocessor real-time scheduling is more promising than laxity under normal load. The work of Bashir, [9] solved the two major round robin' problems (i.e. choosing optimal time quantum (TQ) and context switching). Number of tasks (N) and average burst time of all tasks (ABT) are taken as fuzzy inputs and generate TQ as the output. The TQ to schedule the tasks based on round robin policy to solve the first problem. Another fuzzy Inference system was later introduced with two inputs, Laxity and N, and one output (Preemption Status) to solve the second problem. When his proposed model was compared with standard round robin, result shows that his approach outperformed the existing round robin. Mohammed and Mostafa [10] proposed a fuzzy approach to perform real-time scheduling in which the scheduling parameters are treated as fuzzy variables. They chose priority and laxity as the input linguistic parameters. They also consider another case of priority and deadline. Both cases had runtime priority as the output. Assigning priority to tasks according to their deadlines is simple yet successful strategy for uniprocessor real-time scheduling [10]. The two cases were simulated with some fuzzy rules using centroid defuzzification method of Mamdani inference to generate the output priority. The simulation results show that the output priority based on deadline is much better than the output priority based on laxity, knowing that the initial priority is the same for the two cases.

III. REAL-TIME SCHEDULING ALGORITHMS

There are various standard scheduling algorithms used in CPU scheduling but most of this algorithms are mostly implemented on uniprocessor such as first come first serve (FCFS), shortest job first (SJF), priority and round robin (RR) scheduling algorithms. In this paper we examined two basic real time algorithms, they are rate monotonic (RM) and earliest deadline first (EDF).

i. **Rate monotonic (RM) algorithm:** is a uniprocessor static-priority preemptive scheme. The algorithm is static-priority in the sense that all priorities are determined for all instances of tasks before runtime [11]. The priority of a task is been determined by the length of it period. Tasks with short period times are assigned higher priority. RM is used to schedule periodic tasks. The following are preconditions for the rate monotonic algorithm formalized by Liu and Layland [12].
   a. Periodic tasks have constant known execution times and are ready for execution at the beginning of each period (T).
   b. Deadlines (D) for tasks are at the end of each period: (D = T)
   c. The tasks are independent, that is, there is no precedence between tasks and they do not block each other.
   d. Scheduling overhead due to context switches and swapping are assumed to be zero.

ii. **Earliest deadline First (EDF):** is a dynamic priority driven scheduling algorithm which gives tasks priority based on deadline [10]. Some of the preconditions for RM are also valid for EDF. The task with the currently earliest deadline during runtime is assigned the highest priority. That is if a task is executing with the highest priority and another task with an earlier deadline becomes ready it receives the highest priority and therefore preempts the currently running task and begins to execute.

EDF is an optimal dynamic priority driven scheduling algorithm with preemption for a real-time system on a uniprocessor. EDF is capable of achieving full processor utilization [12].

IV. FUZZY INFERENCE SYSTEM

A Fuzzy Inference System (FIS) derive answers from a knowledgebase by using a fuzzy inference engine. This engine provides the methodologies for reasoning around the information in the knowledgebase and results formulations. Fuzzy logic deals with the concept of partial truth which denotes the extent to which a proposition is true. In classical logic everything can be expressed in binary terms (0 or 1, black or white, yes or no). Fuzzy logic replaces Boolean truth values with the truth's degree. Truth's degree is often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in uncertain and imprecise environment. The membership function of a fuzzy set is analogous to the indicator function of the classical sets. Curves are used to express the membership functions [13]. Curve shape defines how each point in the input space is mapped to a membership value or a truth's degree between 0 and 1. Fuzzy Inference Systems (FIS) consists of three stages namely input, processing and output. In input stage the task parameters (such as deadline, execution time and arrival time) are mapped to the appropriate membership functions and truth values.

In processing stage each appropriate rule is invoked and each of them generates a result [14]. The results of the rules are then combined. Finally, in the output stage the combined result is converted back into a specific output value [15]. The processing stage, called the inference engine, works with the help of a collection of logic rules in the form of IF-THEN statements, where the IF part is called the "antecedent" and the THEN part is called the "consequent". Fuzzy inference systems have several rules, Knowledgebase stores these rules. An example of fuzzy IF-THEN rules is: IF Deadline is critical then priority is high, in which Deadline and priority are linguistics variables. There are two common inference processes [15]. First is called Mamdani's fuzzy inference method proposed by Ebrahim Mamdani [16] in 1975 and the other is Takagi-Sugeno-Kang, method of fuzzy inference introduced in 1985 [17]. These two methods are the same in many respects, such as the procedure of fuzzifying the inputs and fuzzy operators. The main difference between these two methods is that the Sugeno output membership functions are either linear or constant but Mamdani's inference expects the output membership functions to be fuzzy sets. In this paper Sugeno method was used.

V. THE PROPOSED (NFSA) MODEL

The proposed model; new fuzzy scheduling algorithm (NFSA) consists of three inputs scheduling parameters (arrival time, computation time and deadline) as shown in fig.1.

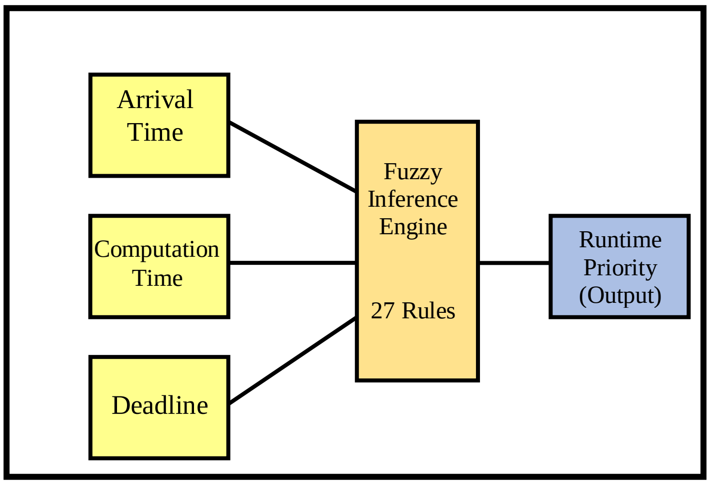

Fig. 1: The proposed NFSA Inference Model

These scheduling parameters were considered because they could guarantee scheduling fairness. The output of the system is the runtime priority which determines the order of tasks execution in a global ready (internal priority) queue. Fuzzy rules combine these parameters as they are connected in real worlds.

The input variables were mapped into the fuzzy sets as illustrated in fig. 2, fig.3 and fig. 4 respectively. The triangular shape for the membership function was used for each linguistic term. It is very difficult for the expert to adjust these membership functions in an optimal way. However, there are some techniques for adjusting membership functions. Those techniques were not considered in this research work. They can be further studied in the future work.

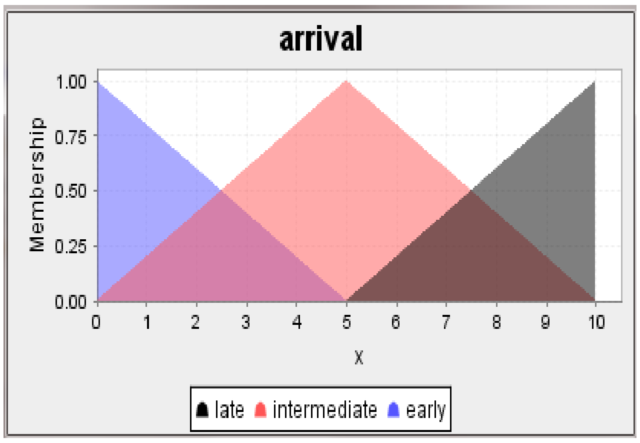

Fig. 2: Membership Function for Arrival Time

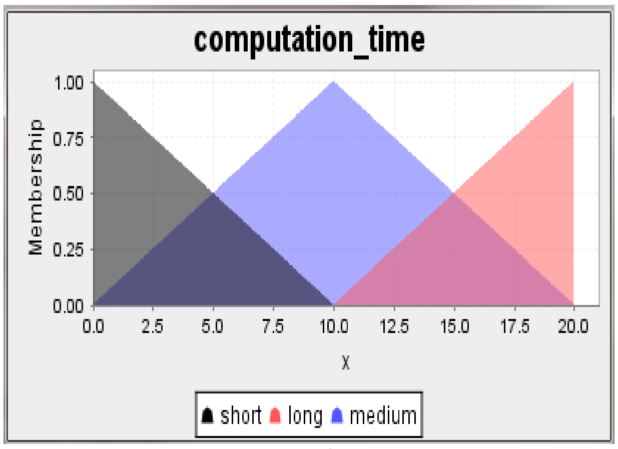

Fig. 3: Membership Function for Computation-Time

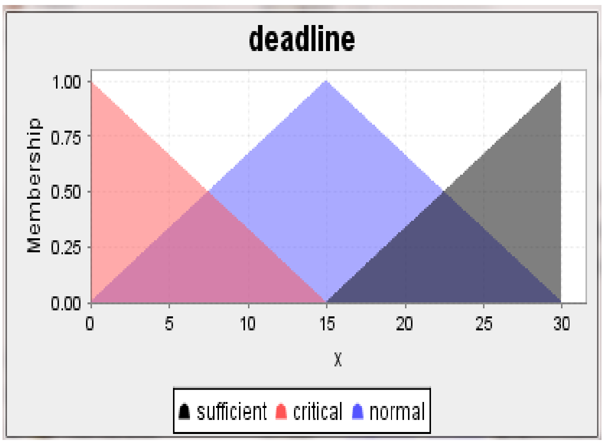

Fig. 4: Membership Function for Deadline

This research work consists of twenty-seven fuzzy rules but few of these rules are mentioned here:

- If (arrival is early) AND (computation_time is short) AND (deadline is critical) THEN (priority is high)
- If (arrival is intermediate) AND (computation_time is short) AND (deadline is sufficient) THEN (priority is normal)
- If (arrival is early) AND (computation_time is long) AND (deadline is sufficient) THEN (priority is low)
- If (arrival is late) AND (computation_time is short) AND (deadline is critical) THEN (priority is high)

However, for the EFA algorithm, the arrival-time parameter was removed and the computation-time was replaced with External priority. Both algorithms has the procedure in the design of their membership function and the output (runtime priority).

VI. THE PROPOSED ALGORITHM

NFSA Algorithm

The Fuzzy Inference Scheduler do the followings as depicted in fig. 5:

```
Loop
  i.   Initialize a task pool $N$ in an arrival queue with task parameter $T_i(a_i, c_i, d_i)$ at $\lambda_0$.
       Where $(i=1,2,3,.....,n)$.
  ii.  For each ready task, feed in task parameter $T_i(a_i, c_i, d_i)$ into fuzzy inference engine at $\lambda_1$.
       Consider the fuzzy inference engine output as the *runtime priority* $r_i$ for each task execution.
  iii. Sort all tasks $T_i$ in descending order of *runtime priority* $r_i$ into the priority (ready) queue at $\lambda_2$.
  iv.  Since all processors are idle at the initial stage,
         • Assign processor $P_{i(i:1,2,3,...,m)}$ to the first set of tasks $T_{i(i:1,2,3,...,m)}$
           with highest *runtime priority* $r_i$ and execute at $\lambda_3$. For $(n \geq m)$.
         • Search through all the processor $P_{i(i:1,2,3,...,m)}$ with the least computation weight $(cw)$.
       If $cw(P_i) < cw(P_{(i+1)})$,
       Assign $P_i$ to $T_{(m+1)}$ and execute
       Else,
       Assign $P_{(i+1)}$ to $T_{(m+1)}$ and execute.
  v.   Update the system states.
       End Loop.
```

VII. NEW FUZZY SCHEDULER ARCHITECTURE

As illustrated in fig. 5, the new fuzzy scheduler loads the set of tasks from the arrival queue into fuzzy inference engine by fuzzifying each task parameters. The fuzzy inference engine then applied fuzzy (AND) operators, fuzzy rules stored in the knowledge base and implication methods to generate the aggregate values which are finally defuzzified as output (runtime priority). The Scheduler follows the NFSA algorithm to schedule the task and execute in the order of internal/runtime priority

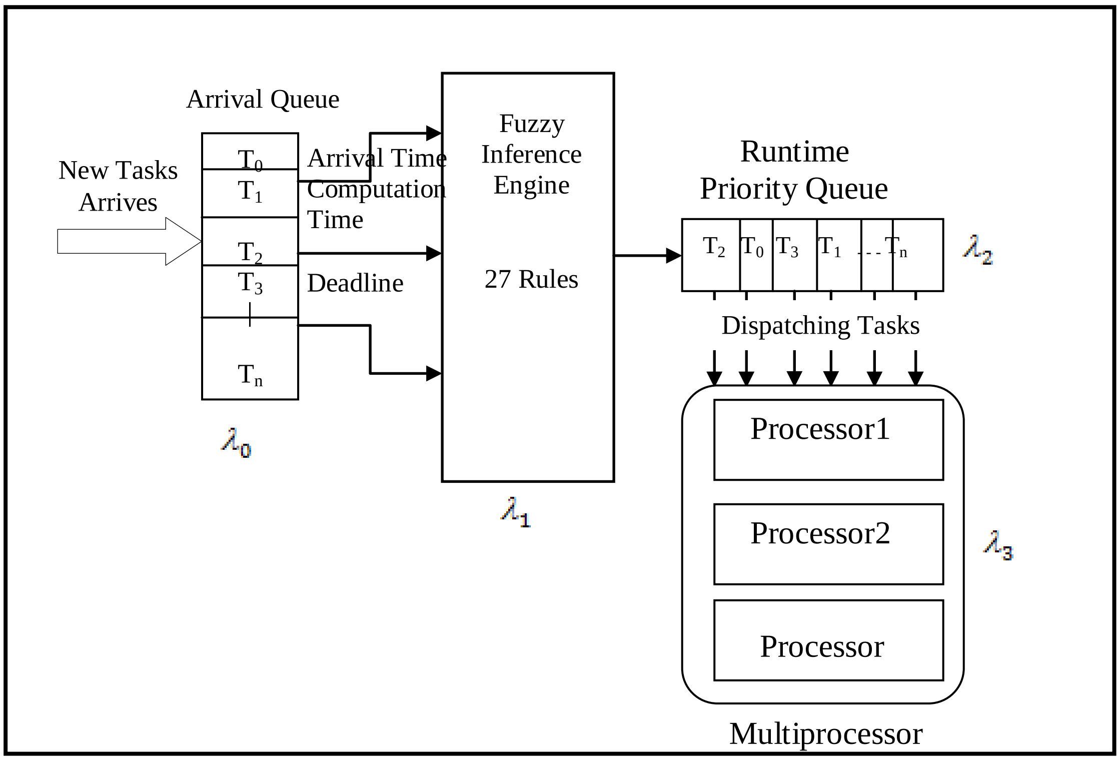

Fig 5: The Architectural View of the Proposed Model (NFSA)

VIII. CASE ASSUMPTION

In a real-time processing environment, tasks of different characteristics are submitted to the multiprocessor by the fuzzy scheduler as described in the previously, this research simulate a fuzzy system comprising of 5 to 25000 real-time tasks, which were assigned to multiprocessor based on the EFA and NFSA algorithm. In order to facilitate the feasibility analysis of this research, the following assumptions were made:

Let T_i represents a periodic task, and U represents a set of periodic tasks;

i. A task cannot suspend itself (i.e. no pre-emption).

ii. All tasks T_i are independent (i.e. there is no relation between the tasks from the same set U).

iii. All tasks T_i have deadline D_i equal to their next request time (period), [12].

iv. All the processors are identical.

v. All tasks T_i evacuate the arrival queue at the same time into fuzzy inference engine.

vi. All tasks T_i are activated in the *runtime priority* (ready) *queue*. Therefore, all tasks in *runtime priority queue* arrive at the processors node at time t = 0.0ms

IX. RESULTS AND PERFORMANCE EVALUATION

The performance of NFSA was compared with EFA which only consists of external priority and deadline as it scheduling parameters. The performance metrics used were carefully chosen in order to reflect the real characteristics of a real-time system. As stated in the previous sections, the performance metric considered are; average turnaround time (ATAT), average response time (ART) and number of deadline missed which is an influential metric in scheduling algorithms for soft real-time systems.

However, the numbers of processors considered in this research are 3, 10 and 100 as the multiprocessor for the simulation. Computation time ranges from 1 – 25ms were applied across the processors and 5 – 25000 tasks were randomly generated with different load factor. The task parameters arrival time was generated using Poisson distribution while computation time and deadline were generated using uniform and normal distribution. In this research several test cases were simulated and the behaviours of both algorithms were compared with each other to determine the strength of the proposed algorithm.

The proposed NFSA outperformed the EFA as shown in the fig. to 6 to fig. 8 in term of the average response time.

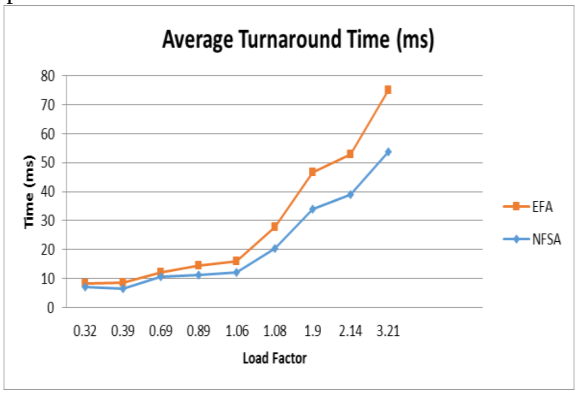

Fig. 6: Average Turnaround Time for 3 processors

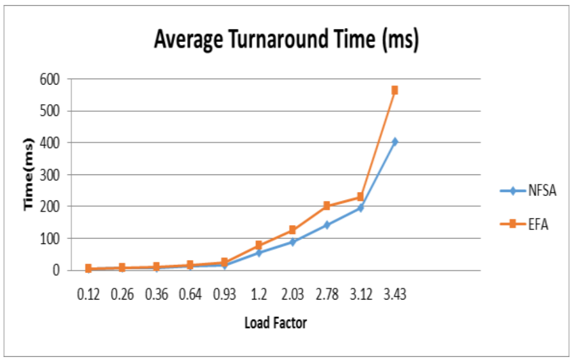

Fig. 7: Average Turnaround Time for 10 processors

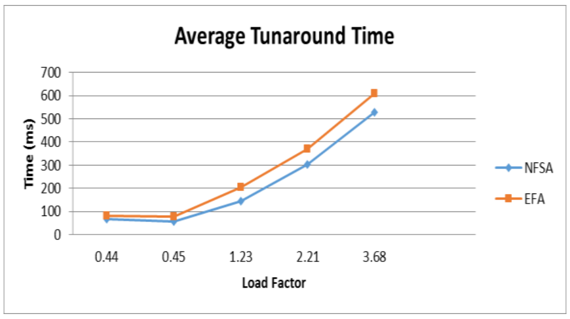

Fig. 8: Average Turnaround Time for 100 processors

As shown in fig.9 to fig. 11, it was observed that the load factor is far less than one (i.e. system under normal load), both algorithms have similar performance for processor 3 and 10. However, as the load factor approaching one and above (when the system becomes overloaded), the response time of NFSA is much tardier than EFA. As the load factor and number of processors increases NSFA algorithm show more better performance. These results have proved that our objectives have been achieved by minimizing the average turnaround time and average response time.

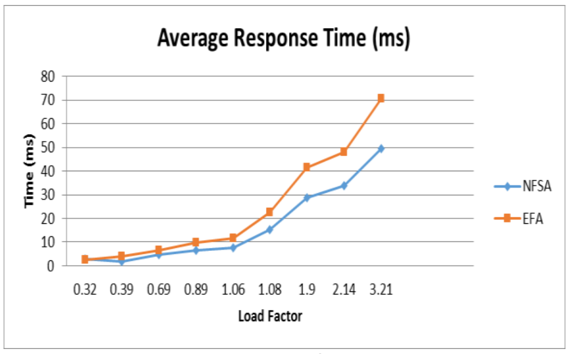

Fig. 9: Average response time for 3 processors

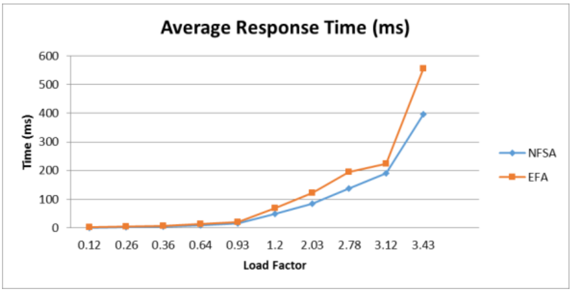

Fig.10: Average response time for 10 processors

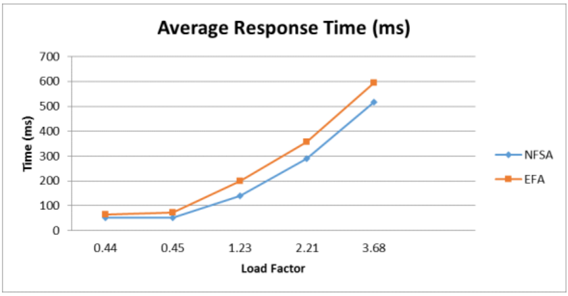

Fig. 11: Average response time 100 processors

When a system load factor is less than one, there is tendecy for all realtime tasks to meet their deadline [12]. Fig. 12 to fig. 14 show that, tasks meet their deadline in both EFA and NFSA when load factor approaching 1 in 3 and 10 processors but as the system load goes beyond 1, NFSA performaed better. In 100 processors, the graph clearly show the performance difference of both algorithms at 0.5 NFSA is tardier in number of deadline missed. Thus, for the correctness of the proposed algorithm, NFSA have shown that deadline missed is minimized on all the three multiprocessors compare to EFA algorithm.

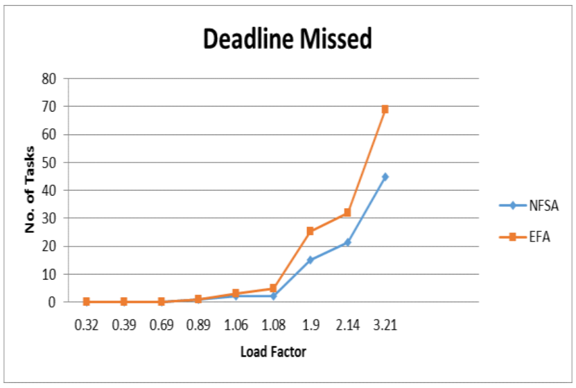

Fig. 12: Deadline missed on 3 processors

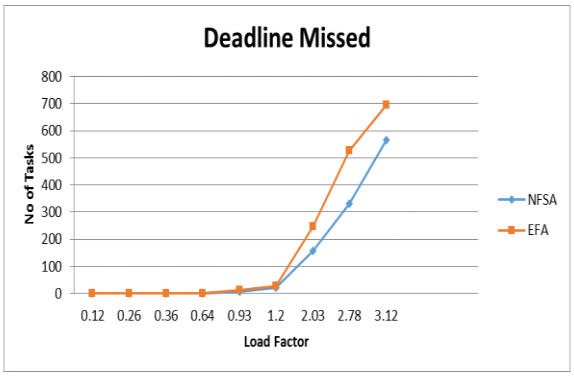

Fig. 13: Deadline missed on 10 processors

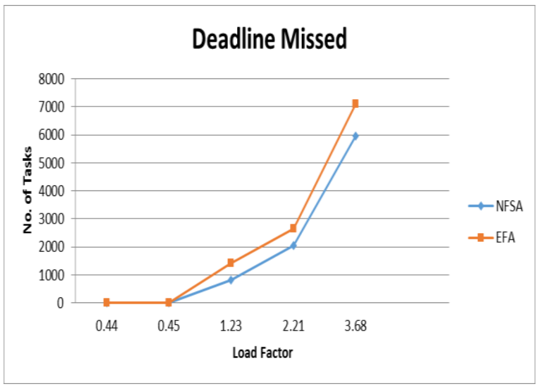

Fig. 14: Deadline missed on 100 processors

X. CONCLUSION AND FUTURE WORKS

In this paper we have successfully mapped fuzzy inference engine on multiprocessor and the results show that, the processors are better utilized thereby improved the system by minimizing the turnaround time, response time and number of deadline missed. In the future, for improving the time complexity of the system, rule reduction techniques will be applied to the system. Also, to improve performance, adjusting membership functions with adaptive methods of inference is required.

ACKNOWLEDGMENT

We wish to acknowledge the contributions and the tireless efforts given to me by my supervisors and my lectures; Prof. Sahalu S. Junaidu and Dr. Saleh E. Abdullahi.

REFERENCES

[1] G. Sagar, A. Neha and D. Kamal, A Fuzzy Approach For Task Scheduling in a Real Time Distributed System. *International Journal of Research in Engineering and Applied Sciences. 2*(2), pp. 1740-1742, ISSN: 2249-3905, 2012.

[2] W. Stallings, Operating Systems Internals and Design Principles. Prentice-Hall, 5th Ed. ISBN:0-13-147954-7, Englewood Cliffs, 2004.

[3] K. Ramamritham and J. A. Stankovic, Scheduling Algorithms and Operating Systems Support for Real-Time Systems. *Institute of Electrical and Electronic Conference*, *82*(1):55-67. Jordan, 1994.

[4] M. Sabeghi, M. Naghibzadeh and T. Taghavi, Scheduling nonpreemptive periodic tasks in soft realtime systems using fuzzy inference. 9th *Institute of Electrical and Electronic Engineers*. International Symposium, ISBN:0-7695-2561-X, doi:10.1109/ISORC.2006.70, Korea, 2006.

[5] B. Shahzad and M. Afzal, Optimized Solution to Shortest Job First by Eliminating the Starvation. *The 6th Jordanian International Electrical and Electronics Engineers Conference. Jordan*, 2006.

[6] D. Sheo, G. Payal and K. Kawaljeet, A Fuzzy Approach Scheduling on More Than One Processor System in Real Time Environment. *International Journal of Scientific Research Engineering & Technology* (IJSRET), *1*(5):289-293, ISSN 2278 – 0882, 2012.

[7] H. Mahdi, M. Sied and L. Caro, Soft Real-Time Fuzzy Task Scheduling for Multiprocessor Systems. *International Journal of Intelligent Technology*. *2*(4): ISSN 1305-6417, 2007.

[8] J. Shatha and A. Kasim, Design and Evaluation of a Fuzzy-Based CPU Scheduling Algorithm. Information processing and Management: 45-52, Springer-*Verlag International Journal*, Berlin, 2010.

[9] A. Bashir**,** Fuzzy Round Robin CPU Scheduling Algorithm. *Journal of Computer Science*, doi:10.3844/jcssp.2013.1079.1085, 1079-1085, 2013.

[10] M. Blej and M. Azizi, Task Parameters Managing and System Accuracy in Fuzzy Realtime Scheduling. International Journal of Engineering Sciences and Research Technology (IJESRT); ISSN:2277-9655, 5(7):60-64, 2016.

[11] J. Strosnider, J. Lehoczky and L. Sha, The Deferrable Server Algorithm for Enhanced Aperiodic Responsiveness in Hard Real-Time Environments. *Institute of Electrical and Electronic Engineers*, Transactions on Computers, *44*(1), 1995.

[12] C. Liu and J. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery* (ACM), *20* (1): 46–61,doi: 10.1145/321738.321743, 1973.

[13] N. Thai, Real-time scheduling in distributed systems. *Parallel Computing in Electrical Engineering, International Conference*, Warsaw, Poland, 165- 170, 2002.

[14] G. William, An Optimization Approach to Employee Scheduling Using Fuzzy Logic. (MSc. Thesis, California Polytechnic State University, San Luis Obispo), 2011.

[15] C. Bindi, Fuzzy Logic Membership Function. Retrieved March 13, 2014, from

http://www.bindichen.co.uk/post/AI/fuzzy-inference-membership-function.html

[16] E. Mamdani and S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*. *7*(1):1-13. 1975.

[17] A. Zadeh, Fuzzy Sets. *Journal of Information and Control*. *Vol 8*,338-353, 1965.