International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Hybrid Scheduler (S_LST) for Soft Real-Time System based on Static and Dynamic Algorithm

Jay Teraiya, Apurva Shah

Abstract*: In the Soft Real-Time System scheduling process with the processor is a critical task. The system schedules the processes on a processor in a time interval, and hence the processes get chance to executes on the processor. Priority-driven scheduling algorithms are sub-categorized into mainly two categories called Static Priority and Dynamic Priority Scheduler. Critical Analysis of more static and dynamic priority scheduling algorithms have been discussed in this paper. This paper has covered the static priority algorithms like Rate Monotonic (RM) and Shortest Job First (SJF) and the dynamic priority algorithms like Earliest Deadline First (EDF) and Least Slack Time First (LST). These all algorithms have been analyzed with preemptive process set and this paper has considered all the process set are periodic. This paper has also proposed a hybrid approach for efficient scheduling. In a critical analysis, it has been observed that while scheduling in underload situation dynamic priority algorithms perform well and even EDF also make sure that all process will meet their deadline. However, in an overload situation, the performance of dynamic priority algorithms reduce quickly, and most of the task will miss its deadline, whereas static priority scheduling algorithms miss a few deadlines, even it is possible to schedule all processes in underload situation, whereas in an overload situation, the static algorithms perform well compared to the dynamic scheduler. This paper is proposing one Hybrid algorithm call S_LST which uses the concept of LST and SJF scheduling algorithm. This algorithm has been applied to the periodic task set, and observations are registered. We have observed the Success Ratio (SR) & Effective CPU Utilization (ECU) and compared all algorithms in the same conditions. It is noted that instead of using LST and SJF as an independent algorithm, Hybrid algorithm S_LST performs well in underload and overload scenario. Practical investigations have been led on a huge dataset. Data Set consists of the 7000+ process set, and each process set has one to nine processes and load varies between 0.5 to 5. It has been tried on 500-time unit to approve the rightness everything being equal.

Keywords: Soft Real-Time System, RTOS, RM, SJF, LST, EDF, S_LST

I. INTRODUCTION

Real-Time Systems has to complete its work and deliver its services on a timely basis. It makes sure that its task will be completed before its deadline. Example of a Real-Time system is vehicle control, flight control, healthcare equipment, and many more. Typical PC run nonreal-time applications such as a browser, editor, different user applications. When the real-time system works correctly, and well, they make us forget their existence [1].

The real-time system is sub-categorized into mainly two types: hard and soft. There are many definitions of hard and soft real-time systems. Real-Time system is considered as Hard if the process fails to meet its deadline, then it will be a fatal fault. In Hard Real-Time, if the process missed its deadline, then result produced by the job after the deadline may have disastrous consequences. A few examples of Hard Real-Time Systems are Metro Train and its signal system, Missile technology, Flight control system. The real-time system is considered as Soft if the late completion of the process is undesirable. However, a few misses of soft deadlines do no serious harm; only the system's performance becomes poor. A few examples of Soft Real-Time systems include ATM System, Mobile application and telephone switches [7].

The real-time system has three kinds of task model call Periodic, Aperiodic and Sporadic tasks. In the periodic task, each task generated at regular time intervals. The Real-Time system is invariably required to respond to external events and to respond; it executes aperiodic or sporadic tasks whose release times are not known to the system in advance. We call the task is aperiodic if the process in it have soft deadlines. Each unit of work is scheduled and executed by the system as a process. Each process has a different characteristic like release time, deadline, period and execution time. The release time of a process is the instant of time at which the job become available for execution. The process can be scheduled and executed at any time after its release. The deadline for a process is the instant of time by which its execution needs to be completed. The deadline for a process sometimes called absolute deadline, which is equal to its release time plus its relative deadline. The execution time of any process is considered as the unit amount of time required for the process to execute it on the processor. If the process is periodic, then the period of the process indicates the occurrence interval of the given process. In RTOS, selecting the scheduling algorithm is a critical task, and it will be decided based on the characteristics of the RTOS and the process type [2]. The scheduler can be divided into two categories, static and dynamic, which depends on the priority they follow in selecting the process for execution.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author
Jay Teraiya*, Department of Computer Engineering, Marwadi University, Rajkot, India. Email: jay.teraiya@gamil.com
Apurva Shah, Department of Computer Science and Engineering The Maharaja Sayajirao University of Baroda, Baroda, India. Email: apurva.shah-cse@msubaroda.ac.in

The static algorithm uses a unique priority to each process throughout the scheduling. Rate Monotonic (RM) and Deadline Monotonic (DM) are an example of static priority algorithms. Dynamic algorithm priority changes during the scheduling process. Earliest Deadline First (EDF) and Least Slack Time First (LST) are an example of dynamic priority algorithms [10][11].

In this paper, we have compared all dominant dynamic and static scheduling algorithms and did their critical analysis. All algorithm has been compared based on Success Ratio (SR) and Effective CPU Utilization (ECU) parameters. This paper also proposed an effective scheduling algorithm call S_LST, which is using characteristics of LST and SJF. The new algorithm also compared with the rest of all algorithms based on SR and ECU parameters. This paper explains the Static and Dynamic Scheduling algorithm in section II. Their critical analysis based on SR and ECU has been described in section III, and a new efficient algorithm call S_LST has been proposed in section IV, and performance of a new algorithm has been compared and discussed in section V, and paper is ended with a brief conclusion in section VI.

II. THE STATIC AND DYNAMIC SCHEDULING ALGORITHMS

Priority-driven scheduling algorithms are online schedulers that schedule the process according to some priority. It does not pre-decide the process; instead of that, it assigns priorities to process when it is ready to execute. The scheduling algorithm will be executed whenever a new process is released, or currently, running process completes its execution. Priority-driven schedulers categorize based on how priority assigned to each process. Priority-driven algorithms are classified in to two categories: Static Priority and Dynamic Priority. A Static Priority algorithm assigns the same priority to all the periodic processes, and it will remain fixed relative to other processes. Whereas dynamic-priority algorithm changes the priority of the process based on the new process arrives or currently running process completes [12][22].

A. Static Scheduling Algorithms

The Rate Monotonic (RM) and the Shortest Job First (SJF) are well known static priority algorithms. The RM assigns the priority to the process based on their period (the frequency of the task when it occurs). The Rate of the process is already known in RTOS for the periodic task. The rate of a process is the inverse of its period, so higher the rate, the priority of the process will be high [6][13][14]. The Shortest Job First (SJF) assigns the priority to the process based on their required execution time. The required execution time of the process is also known in RTOS and process with the shortest execution time will have the highest priority in SJF [13]. By looking at the approach of both algorithms, its ultimate aim is to gain maximum profit or try to meet the maximum deadline of the given processes.

B. Dynamic Scheduling Algorithms

The Earliest Deadline First (EDF) and the Least Slack Time First (LST) are well known dynamic priority algorithms. The EDF assigns the priority to the process based on the absolute deadline. The absolute deadline for each process is already known in RTOS, and the process which has the smallest absolute deadline will consider as highest priority process [8][14]. The LST is another well-known dynamic priority algorithm, and it assigns priority based on the slack time of the given process. The slack value of the process is equal to absolute deadline minus given time t minus remaining execution time x (slack=d-t-x). The algorithm checks the slacks of all the ready process each time a new process is released, or the existing process completes. The process with the smallest slack value will have the highest priority [9][15][16][17]. By looking at the approach of both algorithms, its ultimate aim is to meet the deadline of the given process.

For any set of periodic processes, we can verify its stimulability is possible or not using its occurrence period(T), its execution time(C), and its deadline(D). This ratio U_p is called the utilization factor of the task set as shown in equation 1.

$$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i} \qquad (1)$$

U_p is called the total processor utilization factor and represents the fraction of processor time used by the periodic task set. If U_p>1 no feasible schedule exists for the task set with an algorithm, and it is overload condition.

III. CRITICAL ANALYSIS OF STATIC AND DYNAMIC SCHEDULING ALGORITHM

A. System Consideration and Task Model

In Soft Real Time System, system is already aware with task deadline, its period and the other required data to compute the required time by the task when task is dispatch. The process set is considered pre-emptive. This paper has believed that the system is not having a resource clash problem. Each task in soft real-time systems has a positive value and ultimate goal is to gain maximum value. If a process succeeds, then the system considers its value. If a process fails, then the system gets less benefit from it [18] [19]. In this paper, we have implemented Dynamic and Static scheduling algorithms that apply to the soft real-time system. The value of the task has been considered the same as its computation time required [20].

B. Experimental Environment and Evaluating Parameters

1) Success Ratio (SR):

Success Ratio with real-time systems defined as the ratio of a set of the process which meets their deadline and a total number of process. Success Ration determined with the following equation 2 [21].

$$SR = \frac{Number\ of\ Task\ successfully\ scheduled}{Total\ Number\ of\ Task\ arrived} \qquad (2)$$

2) Effective CPU Utilization (ECU):

Effective CPU Utilization defined as how much CPU time has been utilizing for the processes which can meet their deadline. ECU determined with the following equation 3 [21].

$$ECU = \sum_{i \in R} \frac{V_i}{T} \quad (3)$$

Where,
• V represents process value and,
Process Value = time required to complete the process if the process meets its deadline.
Process Value = 0 if the process does not meet the deadline.
• R is a set of processes, which are scheduled successfully, i.e., completed within their deadline.
• T is the total time of scheduling.

C. Analysis and Observation

RM, SJF, EDF, and LST algorithms are implemented and evaluated with SR and ECU parameters (explained in section 3), and results have been observed. Observation with these results indicates that ECU values persist nearly the same for Dynamic and Static algorithms, but SR values are not 100% with the Static scheduling algorithms. When U_p≤1, it indicates that scheduling of a given task set is possible, but Static scheduling algorithms are failing to schedule all processes, whereas Dynamic scheduling algorithm can schedule this process set. Dynamic scheduling algorithms give optimum results in underload scenario, and it is advisable to use the Dynamic schedulers with underload conditions. In overload situation when U_p>1, observation indicates that Dynamic algorithms performance reduce quickly whereas Static algorithms like RM and SJF are still able to meet a few of their deadline for a given process set. This observation can conclude that in underload EDF and LST give optimal results whereas in overload RM and SJF performed well. Fig. 1 and Fig. 2 provides a graphical representation of results.
The Static and Dynamic algorithms are evaluated here for Soft – RTOS and considering it for a single processor, and pre-emptive process sets and all process set is periodic. All algorithms are evaluated in a similar environment and results have been observed and equated. EDF and LST are dynamic algorithms, and they do well in underload scenario and schedule all processes in a given process set. LST and SJF are static algorithms, and they do well in an overload scenario and try to schedule the maximum process in a given process set. The ideal algorithm can be designed, which uses the features of Dynamic and Static algorithm, and it performs well in underload as well as overload scenario [3][4][5].

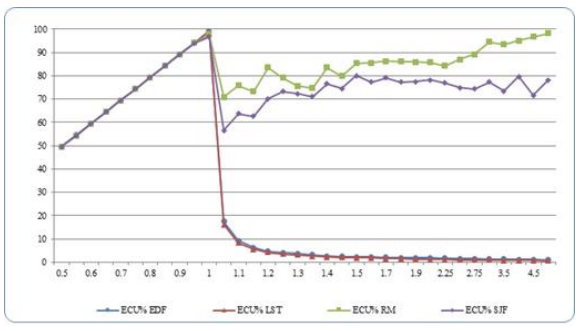

Fig. 1 Load Vs. ECU%

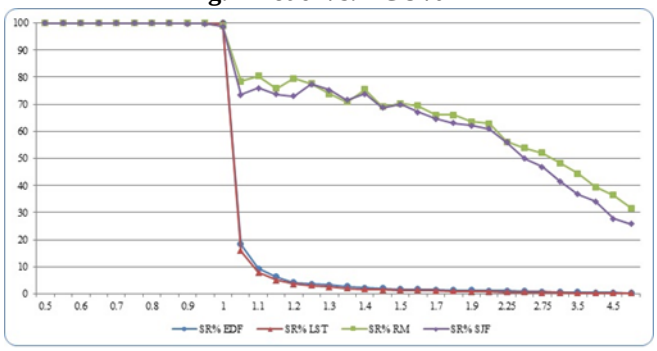

Fig. 2 Load Vs. SR%

IV. THE HYBRID APPROACH FOR EFFICIENT SCHEDULING – S_LST ALGORITHM

S_LST algorithm uses the characteristics of LST and SJF. In underload, situation task priority will be given based on slack time, and in an overload situation, task priority will be assigned based on the shortest execution time. We are considering that the execution time of the task, its arrival time, its period and total CPU load is available with Soft Real-Time System. The scheduling algorithm executes when a currently running task completes or a new task arrives. The algorithm has been described as follows.

S_LST Algorithm for Scheduling

Input: Process Set

Output: MIProcess

1: **if** (Underload Scenario)
2: **for** each process in process set
3: Calculate Slack time for each Process in Process Set
4: Select MIProcess with lowest slack time
5: **end for**
6: **else**
7: **for** each process in process set
8: Calculate Shortest Execution Time for each process
9: Select MIProcess with lowest Execution time
10: **end for**
11: **end if**
12: **return** MIProcess

As shown in Algorithm, when scheduling algorithm invokes; first it observed the CPU load, based on the current process set and available processes are ready for scheduling.

If $U_p < 1$it will assign the task priority based on slack time (Dynamic scheduling algorithm) and if $U_p > 1$ it will assign the task priority based on shortest execution time (Static Scheduling algorithm). The static scheduler aim is to gain maximum profit from the given process set. So, in overload situation where dynamic scheduler performs poorly, SJF algorithm gets more processes meets their deadline.

V. S_LST ALGORITHM RESULTS AND DISCUSSION

Table 1 represents the results of LST, SJF and the S_LST algorithm on the simulator. To evaluate S_LST, we are using a similar environment which we have used to evaluate all Static and Dynamic priority algorithm as per section 3. Table 1 first eleven rows represent the scenario where task set contains 1 to 9 task and Load is less than 1 or equal to 1 ($U_p \leq 1$). Results show that S_LST performs equally well in underload scenarios like LST algorithm in terms of ECU and SR parameter. S_LST uses slack time value of task to assign dynamic priority in underload situation.

Table 1 rest of rows represents the scenario where process set contains 1 to 9 process and Load is greater than 1 ($U_p > 1$). Results show a waste difference in ECU and SR values compare to a simple LST algorithm. When Load is greater than 1, it means that task set is not schedulable, and most of the process misses their deadline with LST algorithm. Table 1 observations reflect that in slightly overload situations LST performance degrades very poorly, whereas SJF able to meet the deadline for few of their process sets. It means in overload situation, SJF gives better performance than LST. That is why S_LST uses static priority in an overload situation. Fig. 3 and Fig. 4 provides a graphical representation of Table 1.

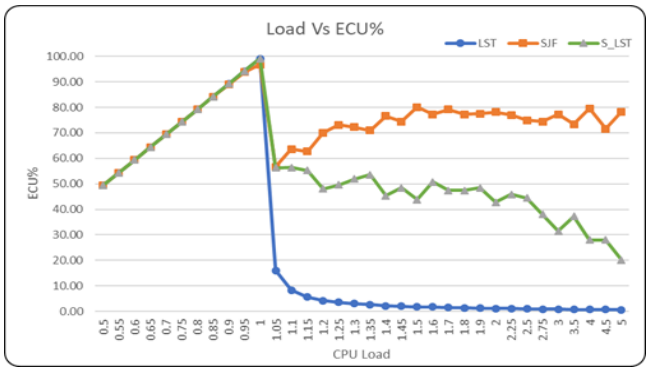

Fig. 3 Load Vs. ECU%

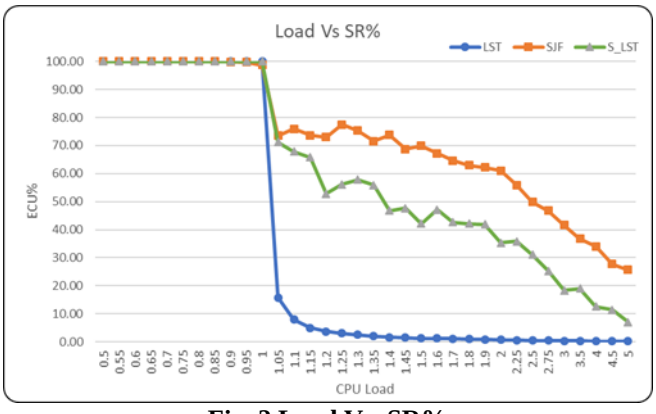

Fig. 2 Load Vs. SR%

Table- I: Comparison of LST, SJF and S_LST

	ECU			SR		
Load	**LST**	**SJF**	**S_LST**	**LST**	**SJF**	**S_LST**
0.5	49.49	49.49	49.49	100.00	100.00	100.00
0.55	54.40	54.31	54.40	100.00	100.00	100.00
0.6	59.39	59.39	59.39	100.00	100.00	100.00
0.65	64.35	64.35	64.35	100.00	100.00	100.00
0.7	69.35	69.35	69.35	100.00	100.00	100.00
0.75	74.31	74.31	74.31	100.00	100.00	100.00
0.8	79.22	79.22	79.22	100.00	100.00	100.00
0.85	84.16	84.15	84.16	100.00	99.99	100.00
0.9	89.16	89.00	89.16	100.00	99.84	100.00
0.95	94.17	93.89	94.17	99.99	99.78	99.99
1	99.10	96.74	99.10	100.00	98.74	100.00
1.05	16.09	56.63	56.63	15.84	73.49	73.49
1.1	8.33	63.60	63.60	7.90	75.98	75.98
1.15	5.58	62.66	62.66	5.06	73.66	73.66
1.2	4.21	70.08	70.08	3.67	73.06	73.06
1.25	3.56	73.20	73.20	3.06	77.47	77.47
1.3	3.09	72.24	72.24	2.53	75.34	75.34
1.35	2.63	70.99	70.99	2.09	71.55	71.55
1.4	2.20	76.57	76.57	1.71	73.80	73.80
1.45	2.01	74.45	74.45	1.52	68.76	68.76
1.5	1.83	80.07	80.07	1.33	69.96	69.96
1.6	1.77	77.26	77.26	1.29	67.20	67.20
1.7	1.58	79.16	79.16	1.07	64.60	64.60
1.8	1.45	77.28	77.28	0.95	63.04	63.04
1.9	1.31	77.53	77.53	0.85	62.21	62.21
2	1.19	78.10	78.10	0.76	61.00	61.00
2.25	1.13	76.95	76.95	0.65	55.91	55.91
2.5	0.98	74.97	74.97	0.54	49.92	49.92
2.75	0.91	74.42	74.42	0.47	46.83	46.83
3	0.86	77.23	77.23	0.40	41.67	41.67
3.5	0.75	73.37	73.37	0.33	36.76	36.76
4	0.73	79.57	79.57	0.27	34.09	34.09
4.5	0.71	71.58	71.58	0.24	27.74	27.74
5	0.66	78.22	78.22	0.20	25.71	25.71

VI. CONCLUSION

The Static Algorithms (RM and SJF) and Dynamic Algorithms (EDF and LST) are implemented for scheduling of soft real-time system with a single processor and pre-emptive task sets and done a critical analysis of these algorithms with ECU and SR parameter in this paper. These algorithms are simulated with periodic task sets; results are obtained and compared. Observation says that dynamic algorithms perform well in underload situations and gives a guarantee to meet all the deadlines of a given process set. In overload (Load is > 1) situation, dynamic algorithms performance degrades very poorly. So, in underload, dynamic algorithms are advisable but not with an overload situation.

Retrieval Number: B3837129219/2019©BEIESP
DOI: 10.35940/ijeat.B3837.129219
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Static algorithms miss few process deadlines even in underload situations. It has been observed that with the specific process set, even it is possible that all processes can meet their deadline, but static algorithms are failed to schedule it. So, in underload, static schedulers are not advisable, but in overload, they perform well compared to dynamic algorithms. Based on this observation we have proposed a hybrid approach for efficient scheduling in Soft Real-Time system call S_LST. S_LST algorithm assigns the static priority in overload situations will perform better in all situations compare to a single approach. Developing a scheduling algorithm using swarm (ACO) has been done for the Soft Real-Time system [21]. There is still multiple research possibility where we can use swarm intelligence techniques like Gravitational Search Algorithm (GSA) or Particle Swarm Optimization (PSO) and can design an efficient scheduling algorithm which can perform well in underload and overload situation.

REFERENCES

1. El Ghor, H., Hage, J., Hamadeh, N., & Chehade, R. H. (2018). Energy-Efficient Real-Time Scheduling Algorithm for Fault-Tolerant Autonomous Systems. Scalable Computing: Practice and Experience, 19(4), 387-400.
2. A. Magdich, Y. Hadj Kacem, M. Kerboeuf, A. Mahfoudhi, and M. Abid, "A design pattern-based approach for automatic choice of semi-partitioned and global scheduling algorithms," Inf. Softw. Technol., vol. 97, no. November 2017, pp. 83–98, 2018.
3. J. Teraiya and A. Shah, "Comparative Study of LST and SJF Scheduling Algorithm in Soft Real-Time System with its Implementation and Analysis," 2018 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2018, pp. 706–711, 2018.
4. J. Teraiya, A. Shah, and E. Foundation, "Analysis of Dynamic and Static Scheduling Algorithms in Soft Real-Time System with its Implementation," Soft-Computing: Theories and Applications (SoCTA - 2018) Jalandhar, India 21-23 December 2018.
5. Konar, D., Bhattacharyya, S., Sharma, K., Sharma, S., & Pradhan, S. R. (2017). An improved Hybrid Quantum-Inspired Genetic Algorithm (HQIGA) for scheduling of real-time task in multiprocessor system. Applied Soft Computing, 53, 296-307.
6. Feld, T., Biondi, A., Davis, R. I., Buttazzo, G., & Slomka, F. (2018). A survey of schedulability analysis techniques for rate-dependent tasks. Journal of Systems and Software, 138, 100-107.
7. Laalaoui, Y., & Bouguila, N. (2014). Pre-run-time scheduling in real-time systems: Current researches and artificial intelligence perspectives. Expert Systems with Applications, 41(5), 2196-2210.
8. Yang, K., & Anderson, J. H. (2015, August). On the soft real-time optimality of global EDF on multiprocessors: From identical to uniform heterogeneous. In 2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications (pp. 1-10). IEEE.
9. Benhai, Z., Yuan, Y., Hongyan, M., Dapeng, Y., & Libo, X. (2016, May). Research on optimal ELSF real-time scheduling algorithm for CPS. In 2016 Chinese Control and Decision Conference (CCDC) (pp. 6867-6871). IEEE.
10. A. Mohammadi and S. G. Akl, "Scheduling Algorithms for Real-Time Systems", in School of Computing, Queen's University, Kingston, Ontario, 2005.
11. Thakor, D., & Shah, A. (2011, December). "D_EDF: An efficient scheduling algorithm for real-time multiprocessor system", in Information and Communication Technologies (WICT), Mumbai, India, pp. 1044-1049, 2011.
12. D. G. Harkut, "Comparison of Different Task Scheduling Algorithms in RTOS : A Survey," vol. 4, no. 7, pp. 1236–1240, 2014
13. Li, W., Kavi, K., &Akl, R. "A non-preemptive scheduling algorithm for soft real-time systems", in Computers & Electrical Engineering, Vol. 33(1), pp. 12-29, 2007.
14. Buttazzo, G. C. "Rate monotonic vs. EDF: judgment day", in Real-Time Systems, Vol. 29(1), pp. 5-26, 2005.
15. M. Patel and B. Oza, "An Improved LLF_ DM Scheduling Algorithm for Periodic Tasks by Reducing Context Switches," in International Journal of Advance Engineering and Research, vol. 2, pp. 248–254, 2015.
16. Belagali, R., Kulkarni, S., Hegde, V., & Mishra, G. "Implementation and validation of dynamic scheduler based on LST on Free RTOS", in Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), Mysore, India, pp. 325-330, 2016, December.
17. Chen, G., & Xie, W. "On a laxity-based real-time scheduling policy for fixed-priority tasks and its non-utilization bound", in Information Science and Technology (ICIST), 2011,Tebessa, Algeria, pp. 7-10, 2011.
18. Locke, C. D. "Best-effort decision making for real-time scheduling" (Ph. D Thesis), Computer Science Department, CMU, 1986.
19. Koren, G., & Shasha, D. "D_over: An Optimal On-Line Scheduling Algorithm for Overloaded Uniprocessor Real-Time Systems" in SIAM Journal on Computing, Vol. 24(2), pp. 318-339, 1995.
20. A Shah, "Adaptive scheduling algorithm for real-time distributed systems", in Biologically-Inspired Techniques for Knowledge Discovery and Data Mining,pp. 236-248, 2014.
21. J. Teraiya, A. Shah, & K. Kotecha, "ACO Based Scheduling Method for Soft RTOS with Simulation and Mathematical Proofs" in International Journal of Innovative Technology and Exploring Engineering, Vol. 8, Issue. 12 pp. 4736-4740, 2019.
22. D. G. Harkut, "Comparison of Different Task Scheduling Algorithms in RTOS : A Survey," vol. 4, no. 7, pp. 1236–1240, 2014.

AUTHORS PROFILE

Jay Teraiya, has completed Bachelor of Engineering from GCET – Vallabh Vidyanagar under S. P. University. He has also completed his M. S. in Software Engineering from BITS Pillani. He is pursuing in Ph. D from the M. S. University of Baroda under the guidance of Dr. Apurva Shah.

Dr. Apurva Shah, Associate Professor and Head of Department (Computer Science and Engineering) in Faculty of Technology, the M. S. University of Baroda Gujarat. He is also director of Computer Center in the University. His area of interest are Real Time System, Artificial intelligence and distributed computing. He has completed his Ph. D. from S. P. University Vallabh Vidyanagar.