IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Mathematical Modelling of Engineering Problems
Vol. 9, No. 5, October, 2022, pp. 1225-1232
Journal homepage: http://iieta.org/journals/mmep

Simulator for Scheduling Real-Time Systems with Reduced Power Consumption

Hakeem Al-Fareed[1], Omar Alghamdi[1], Abdulaziz Alshuraya[1], Majed Alqahtani[1], Saud Alwasfer[1], Ahmed Aljomea[1], Atta-ur Rahman[1*], Sumayh Aljameel[1], Gomathi Krishnasamy[2]

[1] Department of Computer Science (CS), College of Computer Science and Information Technology (CCSIT), Imam Abdulrahman bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia
[2] Department of Computer Information Systems (CIS), College of Computer Science and Information Technology (CCSIT), Imam Abdulrahman bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia

Corresponding Author Email: aaurrahman@iau.edu.sa

https://doi.org/10.18280/mmep.090509

Received: 25 May 2022
Accepted: 2 August 2022

Keywords:
EDF, LJF, RPC, RTS, SRT, slack time, CPU scheduling

ABSTRACT

Optimum resources utilization in computing devices especially power is among the prime areas of research from the very beginning of computer systems. However, its importance in the current era has been significantly increased due to the diverse nature of devices and their real time applications. On the other hand, paradigm is shifting towards sustainable resources that are green/environment friendly (low emission) in nature and produce relatively low energy/power. Real time systems (RTS) are relatively power-hungry due to their time constrained nature. So, there is room to investigate the scheduling algorithms (schedulers) with minimum (low) power consumption. On the other hand simulators are the software that mimic the real time environment for various parameter testing without actual implementation that could be costly as well as complex to build in the beginning. In this study, we are intended to develop a simulator for scheduling Real-Time Systems (RTS) with Reduced Power Consumptions (RPC). That is potentially an environment where various algorithms can be tested over different case studies to examine their performance pertaining RPC for RTS.

1. INTRODUCTION

One of the most important aspects in solving a problem in the right way is the algorithm. The algorithm is used in everything including our real life to solve different kinds of problems we face each day. Also, the algorithms are used in computer purposes to organize and execute the tasks in the right way to get the optimal performance of the system and to prevent the errors to occur while executing the tasks. The algorithm is a set of steps to solve the problem in a proper way to get the optimal desired solution.

Real-time systems are defined as those systems in which the accuracy of the system depends not only on the logical result of the computation but also on the production of results in the specified time. Real-time systems are used in a variety of applications such as critical safety systems, control units in power plants, satellite controllers, command systems, and flight control systems. Real-time systems can be categorized into hard real-time systems and soft real-time systems. In hard real-time systems, the responses must occur within the required deadline. Otherwise, missing the deadline may result in huge losses and dangerous consequences. For example, missile control systems. Soft real-time are those systems where deadlines are important but will still function properly if deadlines are not met because the task can be rescheduled or can be completed after the specified time. For example, multimedia and gaming systems [1]. This paper discusses the most used algorithms in Real-Time systems, which are Rate Monotonic and Earliest Deadline First algorithms as well as an explanation of different scheduling algorithms with their respective pros and cons and suitability towards the nature of real time system like whether it is soft real time, hard real time. The goal of this paper is to analyze and investigate different RTS algorithms [2] and compare their performance including the temperature of the algorithm to find out which one of the algorithms generates less heat. This journal will include 12 RTS algorithms such as Least Slack Time scheduling (LST) [3], Longest Job First scheduling (LJF) [4], Shortest Remaining Time scheduling (SRT) [5], etc. each algorithm focuses on different parameters such as deadline time for least Slack Time and some of the algorithms are focusing on same parameters as arrival time for Earliest Deadline First scheduling (EDF).

Moreover, the significance and purpose of the present study is to determine the right algorithm for real time system which requires minimum/reduced power consumption. So far, in the literature, no experimental work has been conducted to the comprehend the issue at hand. It is mainly because in the real time systems, after the time constraint, the power/energy constraint is critical. It is always desired to come up with the scheduling algorithms/schedulers with minimum power consumption.

The rest of this paper is divided as follows: Section 2 will provide the used algorithm in this study. Section 3 will talk about the methodology of testing these scheduling algorithms. Section 4 will provide the experiment results. Section 5 will provide the discussion on the results. Lastly, section 6 provides conclusion and future work.

2. RELATED WORK

Many research works have been done on scheduling algorithms in real-time systems. The researchers aim to find the optimal algorithm for such systems. This section represents some of these works.

A study [6] discusses the scheduling algorithm in the real-time system in terms of the effect of the quality of the real-time scheduling algorithm on the real-time system throughput capacity, response time, and this paper also discusses the features and performance of the real-time system according to the system environment, splitting the real-time system into single processor scheduling, multiprocessor scheduling, distributed scheduling, real-time scheduling algorithms RMS, EDF, and LLF in a single processor.

A study [7] presents a real-time domain summary in scheduling and operating systems Where four scheduling models are discussed: static scheduling, pre-emptive scheduling with fixed priority, dynamic scheduling, and dynamic scheduling for best effort. operating systems in real-time. The authors [8] focus on making some improvements to Earliest First Deadline (EDF) Algorithms in order to reduce the number of relay tasks in addition to the ability to predict their behavior. The earliest first deadline (EFDF) is known. Displays algorithms at the very least complexity by Performance analysis. Based on the results of the experiment, it was found that the earliest deadline first (EFDF) algorithm reduced the complexity time in older tasks, deadline first (EDF) scheduling algorithm in a real-time system. In a multiprocessor system.

They concluded [9] that the EDF scheduling algorithm is an optimal scheduling algorithm for single processors, but it has received little attention from the industry. Fixed Priority, on the other hand, is relatively popular with many commercial real-time operating systems despite offering lower theoretical schedulable processor utilization.

They [10] presented an optimal real-time scheduling algorithm for multiprocessors, which is not based on time quanta called LLREF designed based on a technique of using the T-L plane abstraction for reasoning about multiprocessor scheduling. It showed that scheduling for multiprocessors can be viewed as repeatedly occurring T-L planes, and correct scheduling on a single T-L plane leads to the optimal solution for all times.

Authors [11] talked about that EDF algorithm schedules real-time tasks are based on their deadlines plus that EDF is widely studied as a dynamic priority-driven scheduling scheme because of its optimality for periodic, aperiodic, and sporadic preemptive tasks, optimality for sporadic non-preemptive tasks, and acceptable performance for periodic and aperiodic non-preemptive tasks.

EDF can achieve the highest possible processor utilization for preemptive tasks. Although finding an optimal schedule for periodic and aperiodic non-preemptive tasks is NP-hard [12, 13]. Experiments [14] show that EDF can achieve very good results even when the system is lightly loaded. When the processor is overloaded (i.e., the combined requirements of pending tasks exceed the system's capabilities), EDF performs poorly. Researchers have proposed several adaptive techniques for dealing with heavily loaded situations, but they all require the detection of the overload condition [15].

Some successful and related real time systems algorithms are discussed subsequently. In this regard, it is also mentioned that how good or bad the algorithms are in terms of power consumption.

2.1 Earliest Deadline First scheduling

Earliest Deadline First (EDF) scheduling focuses on the burst time of the task, and it arranges the priority of the tasks according to the burst time. The lower burst time of the task it will assign a higher priority for it. But there is a drawback of this algorithm if the lower priority tasks with a high burst time they may take too long time to start executing. This algorithm can be preemptive or non-preemptive depends on the nature of the system [16].

2.2 First Come First Serve scheduling

First Come First Serve (FCFS) scheduling focuses on the arrival time of the task only and it arranges the tasks according to the arrival time [17].

2.3 Longest Job First scheduling

Longest Job First (LJF) scheduling focuses on the burst time of the task, and it arranges the priority of the tasks according to the longest burst time. And this algorithm can be preemptive or non-preemptive based on the nature of the system [18].

2.4 Longest Remaining Time scheduling

Longest Remaining Time (LRT) scheduling focuses on the burst time of the task same as LJF, but the work of this scheduling is different from LJF. This scheduling arranges the tasks based on the remaining time of the burst time and it arranges the priority of the tasks according to the longest burst time [19].

2.5 Least Slack Time scheduling

Least Slack Time (LST) scheduling focuses on the slack time of the task, and it arranges the priority of the tasks according to the least slack time [20]. This scheduling is not used often because it behaves like EDF scheduling

2.6 Most Slack Time scheduling

Most Slack Time (MST) scheduling focuses on the slack time of the task same as LST scheduling but this scheduling arranges the tasks according to the longest slack time [21].

2.7 Priority Scheduling

Priority scheduling focuses on the priority of the task, and it arranges the execution of the tasks according to the highest priority first. This algorithm doesn't consider the arrival time and burst time. This algorithm can be used as preemptive or non-preemptive [22].

2.8 Round Robin scheduling

Round Robin (RR) scheduling is a preemptive scheduling, and in this scheduling the tasks are assigned to a time called quantum time. This type of scheduling keeps the tasks in the ready queue to execute them while taking in consideration the time slice in the ready queue [23]. In this type of scheduling the job queue behaves like a circular queue. It is more appropriate for the interactive processing.

2.9 Shortest Remaining Time

Shortest Remaining Time (SRT) scheduling focuses on the burst time of the task, but it the work of the scheduling is different from LRT. This type of scheduling focuses on the remaining time of the burst time, and it arranges the priority of the tasks according to the least remaining time [24]. So with that said, the process with minimum left over time is considered as prior than the one with more left over time.

3. RTS SCHEDULING ALGORITHMS

3.1 Scheduling algorithms for uniprocessor

Real-time systems that used a single processor have various scheduling algorithms. As shown in Figure 1, these algorithms can be classified into static and priority-driven algorithms. The static category involves many algorithms such as Round Robin (RR) in which the processor time is divided equally among the tasks. The other category is priority-driven algorithms, and it is the focus of this section.

Static priority is used to determine base time slice of a process. Dynamic priority is used to select a process to be executed next. Real time priorities are defined only for Real time processes and its value can range from 0 to 99.

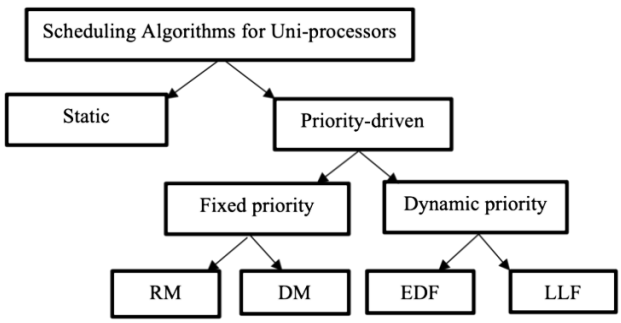

Figure 1. Classification of scheduling algorithms for uniprocessor [11]

3.1.1 Priority-driven scheduling algorithms

As represented in Figure 1, the priority-driven scheduling algorithms are divided into fixed and dynamic. This classification is based on the priority assignment whether the priority is static or changed at running time. This section represents an overview of the most widely used priority-driven algorithms in real-time systems which are EDF and RM. The Rate Monotonic Algorithm is another name for the RM Scheduling Algorithm. The RM algorithm is a fixed or static priority scheduling algorithm. Tasks are preferred by RM based on their period. The disadvantage of this algorithm is that it does not provide a perfect result in a low-load situation. When compared to dynamic scheduling, RM performs better in overloaded situations. In the RM algorithm, the shortest period gives the most chances to execute [12]. The Eq. (1) used for RM is:

$$\frac{C_1}{T_1} + \cdots .. + \frac{C_n}{T_n} \leq U(n) = n(2^{\frac{1}{2}} - 1) \qquad (1)$$

where, C_i stands for computation time, T_i stands for period time, and $U_{(n)}$ stands for CPU utilization [13]. RM can be implemented in any operating system which supports static priority scheme, like VxWorks, DSP/BIOS.

The Earliest Deadline First scheduling algorithm is also known as the nearest deadline first scheduling algorithm. The EDF algorithm is a dynamic scheduling algorithm. The task must be completed as soon as possible. The task with the earliest deadline has the highest priority. EDF Scheduling provides 100 percent task utilization under loaded conditions or when the utilization is less than or equal to 1. In contrast, when task utilization is more than the cross-load factor or slightly overloaded, the utilization of the processor decreases exponentially [12]. The equation used for EDF as follows $T_1 < T_2 < \ldots < T_n$. T stands for task, which the finish time of the current task is less than the next task. EDF is used in real-time operating systems to arrange the processes in a priority queue according to finish time. Table 1 shows the advantages and disadvantages of RM and EDF. Figure 2 represents a case study of how RM and EDF behave on the same task set. The goal of the used task in this case study is to find out which one of the algorithms is faster than the other, and how long the execution time of each task. This task set is just a sample to make the picture of flow work of RM and EDF, but in real life where the RM and EDF used on real-time systems the tasks will become more complex. Suppose a task set consists of three tasks where each task *Ti* is represented by its computation time and the period, Ci and Pi, respectively.

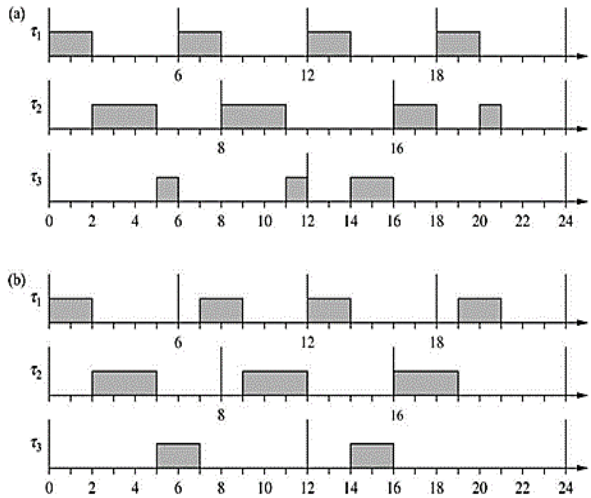

Figure 2. (a) RM and (b) EDF scheduling comparison [14]

The tasks are T_1 (2,6), T_2 (3,8), T_3 (2,12). As it is shown in Figure 2 (a), the priority in RM is assigned based on the period. So, the task with the lowest period has the highest priority. In Figure 2 (b), the EDF's priority is changed based on the task deadline. So, the task with the shortest deadline at each time interval has the highest priority. Figure 3 represents another case study of how RM misbehaves in some conditions. Suppose a task set consists of two tasks T_1 (2,5), T_2 (4,7).

As it is shown in Figure 3 (a), since T_1 has a higher priority than T_2, T_1 will preempt every instance of T_2, and sometimes it may cause a deadline missing. In contrast, in Figure 3(b), the EDF's can schedule this task set because it doesn't cause any deadline missing. As stated, "For larger task sets, the number of preemptions caused by RM increases, thus the overhead due to the context switch time is higher under RM than EDF" [13].

Table 1. RM and EDF advantages and disadvantages

Algorithms	Advantages	Disadvantages
RM (Rate Monotonic)	· Simple to implement. · Commonly used algorithm.	· Waste CPU utilization · Difficult implantation
EDF (Earliest Deadline First)	· Full process utilization	· Misbehave in overloaded conditions

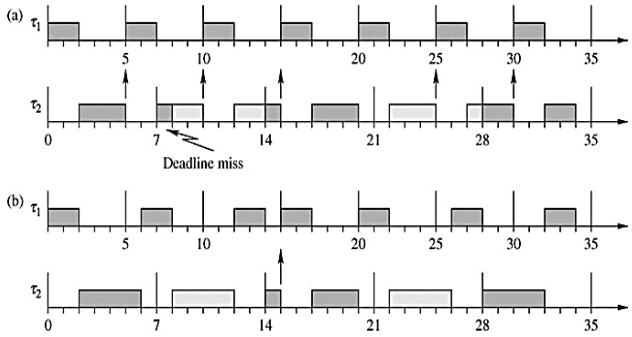

Figure 3. (a) RM, (b) EDF scheduling comparison [14]

3.2 Scheduling algorithms for multiprocessor

As time goes, the need for more than one processor is increased to perform more complex and heavier computations. Multiprocessor systems require a different scheduling scheme than uniprocessor. Many research works have been done in this field to obtain the best scheduling algorithm. Figure 4 represents the algorithm's classification of multiprocessor systems. They are divided into classic and heuristic and the evolutionary algorithms. In the classic category, most algorithms are not exclusively created to be used in multiprocessor environments, however, they achieve less time complexity in multiprocessor systems compared to other categories. One drawback of classic algorithms is that they don't guarantee an optimal solution. The other category is heuristic & evolutionary algorithms, which achieve a near-optimal solution but with more running time.

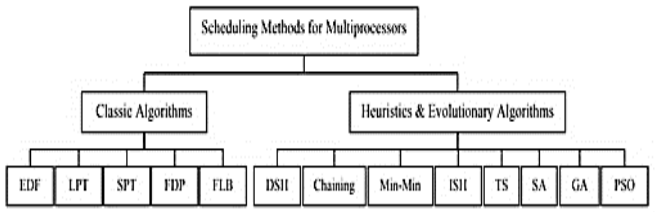

Figure 4. Scheduling algorithms for multiprocessor systems [15]

In Table 2, we present a comparison of some uniprocessor and multiprocessor scheduling algorithms and compare them from different metrics such as priority, CPU utilization, number of contexts switching, optimality, deadline miss chances, response time, predictability, effectiveness, suitability, and limitations.

Table 2. Comparison between uniprocessor and multiprocessor algorithms [16]

Algorithm / Performance Metric	Uniprocessor Algorithms			Multiprocessor Algorithms				
Priority	EDFD	LLFD	MUF Hybrid	IUFD	EDFZLD	ILLFD	MMUF Hybrid	MIUF D
CPU Utilization	High	High	High	High	High	High	High	High
No. of context switching	Less	High	High	High	Very less	Less	Less	Less
Optimal	Yes	Yes	For critical tasks	Yes	No	Yes	Yes	Yes
Deadline miss chances	Average	Average	Less	Less	Less	Less	Less	Very less
Response time	High	Average	Low	High	Low	High	Average	Low
Predictability	Not predictable	Not predictable	Predictable under transient load	Dynamic predictability	More predictable than EDF	More predictable	Predictable under transient load	Dynamic predictability
Effectiveness	Optimal, easy to implement	Takes execution time into consideration	Work in transient overload	Maximize utilization in bound of schedule	Context switching overhead is low	Less context switching	Optimal for non-critical tasks	Improves context switching, response time and CPU utilization
Limitations	Not work in overload, not optimal for pro>1	In laxity time, more context switches occur.	Non-critical task may miss deadline	Context switching is very high	Chances of deadline miss of the critical tasks	Execution time is more.	Only consider static utilization of task set	

4. METHODOLOGY

In this study we will use different RTS algorithms to compare the results. And these RTS algorithms will be used in this test to find out which one of these is the best, because each algorithm works in different way as mentioned in section 2. Some of the algorithms divides into two types of scheduling which are preemptive and non-preemptive. We will take in consideration if the algorithm has these two types or not in testing them. While we are preparing the test of these scheduling algorithms, we take in consideration only one performance aspect of the algorithm which is the generated heat caused by the algorithm or in other words the temperature of the CPU. The dataset used was created by us. And this dataset contains 1,000 of the processes with random arrival time, burst time, slack time, priority, and fixed quantum time which is 8 ns. A standard dataset was used to figure out the test and the optimum analysis [25-28].

In this test we are going into four stages because we will modify the consumed power by the CPU to figure out which algorithm is the best in all stages and which stage is the best. The best algorithm will be the least temperature of the CPU, the average algorithm will be calculated by summation of best and worst temperature dividing by 2 and taking the nearest temperature of the result, and the worst algorithm will be the maximum temperature reached by the CPU.

The CPU was used in this test is Intel I7-9750H 6-core and 32 GB of RAM. Also, the chosen level of under-volting was the optimum level of the used device, because if we under-volted the device more than this level we will face issue in in stability of the operating system. The experiments were conducted at standard room temperature. Furthermore, we will describe now the four stages of the test.

5. EXPERIMENT RESULTS

5.1 Stage 1

In the first stage we made the test with the default power consumption of the CPU which the default watts of the CPU are 90 without under-volting it. And in this stage, we found the best algorithm was Round Robin because it has the lowest CPU temperature which is 46.2℃, the average algorithm was Most Slack Time which the temperature of the CPU reached 48.8℃. Finally, the worst algorithm in this stage was EDF Preemptive which the temperature of the CPU reached 51.8℃. This is shown in Figure 5.

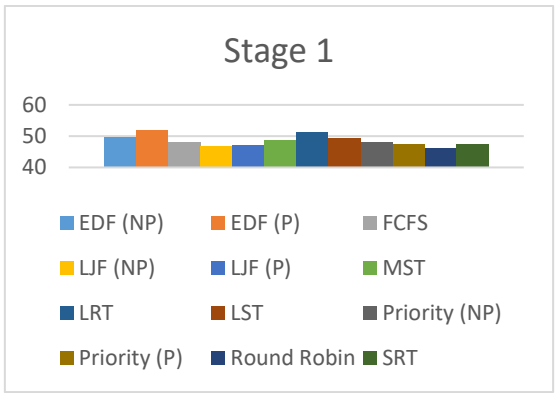

Figure 5. Stage 1

5.2 Stage 2

In the second stage we undervolted the CPU to -100 mv with default watts and found in this stage the best algorithm is LJF non-preemptive which the temperature of the CPU reached 45.5℃. Secondly, the average algorithm was Priority non-preemptive which the temperature of the CPU reached 48.14℃. Finally, the worst algorithm we found was LJF preemptive which the temperature of the CPU reached 50.85℃. This is shown in Figure 6 as the stage two experiment outcome.

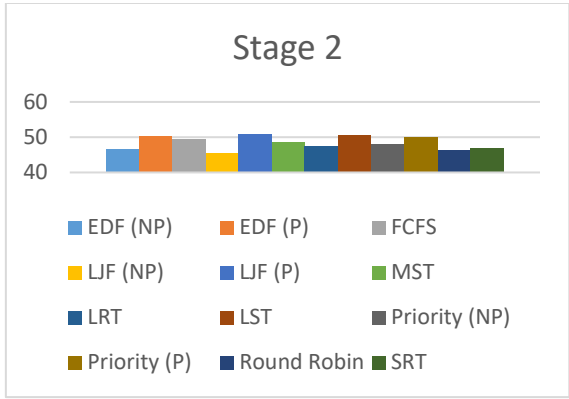

Figure 6. Stage 2

5.3 Stage 3

In the third stage we modified the consumed watts by the CPU to 45 watts without undervolting it. And we found in this stage the best algorithm was Round Robin and LJF non-preemptive because the temperature of the CPU reached 46.28℃. Secondly, the average algorithm we found was Most Slack Time because the temperature of the CPU reached 49.42℃. Finally, the worst algorithm we found was Priority non-preemptive because the temperature of the CPU reached 53.28℃. This is depicted in Figure 7 as stage three test.

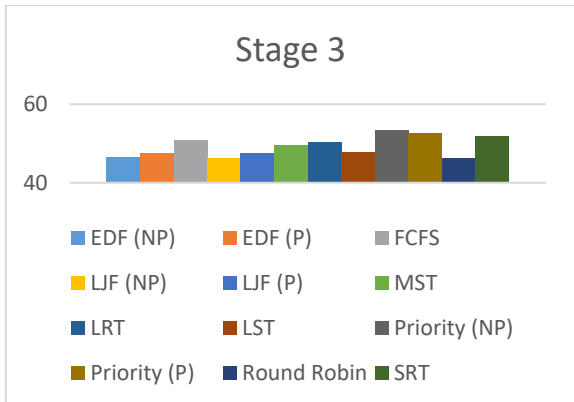

Figure 7. Stage 3

5.4 Stage 4

In the fourth stage we modified the consumed watts by the CPU to 45 and we undervolted it to -100mv and we found in this stage the best algorithm was LJF and EDF both non-preemptive which the CPU temperature reached 45.71℃. Secondly, the average algorithm was EDF pre-emptive which the temperature of the CPU reached 49.14℃. Finally, the

worst algorithm was Priority pre-emptive which the temperature of the CPU reached 53.28℃. This is demonstrated in Figure 8 as stage 4 analysis.

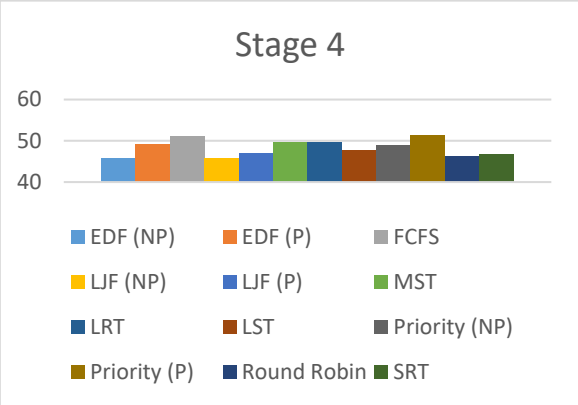

Figure 8. Stage 4

6. RESULTS DISCUSSION

When we finished from the experiment, we found different scheduling algorithms are best in different stages. We found the best algorithm in stage 1 was Round Robin (Figure 1), in stage 2 we found the best algorithm was LJF non-preemptive (Figure 4), in stage 3 we found the best algorithms are Round Robin and LJF non-preemptive (Figure 6.

Finally, in stage 4 we found that the best algorithms are LJF and EDF both non-preemptive (Figure 7). And we can conclude from this experiment adjusting the voltage was having a big impact on the temperature of the CPU. Also, we conclude that the best algorithms are Round Robin and LJF because they are the lowest algorithms in heat generation. And for the stages we found that the best stage was stage 2 and the CPU was under-volted to -100 mv.

7. CONCLUSION AND FUTURE WORK

In this research work we investigate the least generation of heat in CPU scheduling algorithms in different conditions such as minimum average waiting time, improved throughput and reduce power consumption etc. Firstly, we examine the best algorithm in these conditions by various experiments and the results show that the best algorithms were Round Robin and Longest Job First scheduling in a non-preemptive scenario. That is obviously a typical case of real-time systems.

Secondly, the other algorithms need to be improved to generate less heat, because each one of the algorithms is used in different circumstances, and some of the circumstances takes too long to finish the executing of the tasks which the temperature of the CPU will be raised as long as the execution is running, and this will lead to the device failure. In future, machine learning based algorithms may also be investigated in the real time systems especially where the cloud systems are involved, and green technology is evolved to save the power consumed and reduce the carbon discharge from the cloud centers that are mainly depending on the nature of algorithms being used not only the type of high-performance hardware [29-60].

REFERENCES

[1] Bernat, G., Burns, A., Llamosí, A. (2001). Weakly hard real-time systems. IEEE Transactions on Computers, 50(4): 308-321. https://doi.org/10.1109/12.919277

[2] Salam, A., Abbas, S., Khan, Y. (2019). Developing the best scheduling algorithm from existing algorithms for real time operating systems. SSRN Electronic Journal. http://dx.doi.org/10.2139/ssrn.3331997

[3] Abdullahi, I. (2012). Process Scheduling in Longest Job First (LJF) Algorithm. A proposed framework for starvation problem. ResearchGate, Nigeria. http://dx.doi.org/10.13140/RG.2.1.2706.9607

[4] Hefetz, N., Adiri I. (1982). An efficient optimal algorithm for the two-machines unit-time jobshop schedule-length problem. Mathematics of Operations Research, 3(7): 354-360. https://doi.org/10.1287/moor.7.3.354

[5] Hwang, M., Choi, D., Kim, P. (2011). Least slack time rate first: An efficient scheduling algorithm for pervasive computing environment. Journal of Universal Computer Science, 17(6): 912-925.

[6] Li, J., Guo, R.F., Shao, Z.X. (2010). The research of scheduling algorithms in real-time system. In CCTAE 2010 - 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering, 1: 333-336. https://doi.org/10.1109/CCTAE.2010.5544771

[7] Ramamritham, K. Stankovic, J.A. (1994). Scheduling algorithms and operating systems support for real-time systems. In Proceedings of the IEEE, 82(1): 55-67. https://doi.org/10.1109/5.259426

[8] Singh, J. An algorithm to reduce the time complexity of earliest deadline first scheduling algorithm in real-time system. re, accessed on September 10, 2022.

[9] Lunniss, W., Altmeyer, S., Davis, R.I. (2014). A comparison between fixed priority and EDF scheduling accounting for cache related pre-emption delays. Leibniz Transactions on Embedded Systems, 01: 1-24. https://doi.org/10.4230/LITES-v001-i001-a001

[10] Cho, H., Ravindran, B., Jensen, E.D. (2006). An optimal real-time scheduling algorithm for multiprocessors. 2006 27th IEEE International Real-Time Systems Symposium (RTSS'06), pp. 101-110. https://doi.org/10.1109/RTSS.2006.10

[11] Jeffay, K., Stanat, D.F., Martel, C.U. (1991). On non-preemptive scheduling of periodic and sporadic tasks. In Proceedings - Real-Time Systems Symposium, pp. 129-139. https://doi.org/10.1109/real.1991.160366

[12] Garey, M.R., Johnson, D.S. Computers and intractability: A guide to the theory of NP-completeness. SIAM Review, 24(1): 90-91. https://doi.org/10.1137/1024022

[13] Sha, L., Klein, M.H., Goodenough, J.B. (1991). Rate monotonic analysis for real-time systems. Technical Report Submitted to Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, pp. 1-29. https://resources.sei.cmu.edu/asset_files/TechnicalReport/1991_005_001_15923.pdf.

[14] Locke, D. (2022). Best-effort decision making for real-time scheduling. https://.researchgate.net/publication/238690426, accessed on September 11, 2022.

[15] Georges, L., Mühlethaler, P., Rivierre, N. (2000). A few results on non-preemptive real time scheduling. [Research Report] RR-3926, INRIA.
[16] Harkut, D.G., Ali, M.S., Lohiya, P. (2013). Real-time scheduler for wireless sensor network : A review. International Journal of Engineering Research & Technology (IJERT), 2(12): 254-259.
[17] Thakar, H. (2016). Comparison between EDF_RM and EDF_DM in dynamic scheduling algorithm with sporadic task. https://www.academia.edu/73284340/Comparison_between_EDF_RM_and_EDF_DM_in_dynamic_scheduling_algorithm_with_sporadic_task, accessed on September 11, 2022.
[18] Buttazzo, G.C. (2005). Rate Monotonic vs. EDF: Judgment Day. Real-Time Systems, 29: 5-26. https://doi.org/10.1023/B:TIME.0000048932.30002.d9
[19] Rouhifar M., Ravanmehr, R. (2015). A survey on scheduling approaches for hard real-time systems. International Journal of Computer Applications, 131(17): 41-48. https://doi.org/10.5120/ijca2015907656
[20] Pandit S., Shedge, R. (2013). Survey of real time scheduling algorithms. IOSR Journal of Computer Engineering, 13(2): 44-51. http://dx.doi.org/10.9790/0661-1324451
[21] Golconda, K.S., Doğan, A., Özgüner, F. (2004). Static mapping heuristics for tasks with aard deadlines in real-time heterogeneous systems. In: Aykanat, C., Dayar, T., Körpeoğlu, İ. (eds) Computer and Information Sciences - ISCIS 2004. ISCIS 2004. Lecture Notes in Computer Science, vol 3280. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30182-0_83
[22] Burns, A. (1993). Preemptive priority based scheduling: An appropriate engineering approach. Citeseer.
[23] Yaashuwanth, C., Ramadoss, R. (2010). Intellgent time slice for round robin in real time operating systems. pp. 126-131.
[24] Cheriere, N., Bouillud, P.D., Ibrahim, S., Simonin, M. (2016). On the usability of shortest remaining time first policy in shared Hadoop clusters. Proceedings of the 31 st Annual ACM Symposium on Applied Computing, pp. 426-431. https://doi.org/10.1145/2851613.2851626
[25] Sha, L., Abdelzaher, T., årzén, K.E. et al. (2004). Real time scheduling theory: A historical perspective. Real-Time Systems, 28: 101-155. https://doi.org/10.1023/B:TIME.0000045315.61234.1e
[26] Hwang, M., Choi, D., Kim, P. (2011). Least slack time rate first: An efficient scheduling algorithm for pervasive computing environment. Journal of Universal Computer Science, 17(6): 912-925.
[27] Ahmad, F., Mahmud, S.A., Khan, G.M., Yousaf, F.Z. (2013). Shortest remaining processing time based schedulers for reduction of traffic congestion. 2nd International Conference on Connected Vehicles & Expo (ICCVE), at: Las Vegas, USA. http://dx.doi.org/10.1109/ICCVE.2013.6799805
[28] AlKhulaifi, D., AlQahtani, M., AlSadeq, Z., Rahman, A. (2022). An overview of self-adaptive differential evolution algorithms with mutation strategy. Mathematical Modelling of Engineering Problems, 9(4): 1017-1024.
[29] Rahman, A. (2020). GRBF-NN based ambient aware realtime adaptive communication in DVB-S2. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-020-02174-w
[30] Alhaidari, F., Rahman, A., Zagrouba, R. (2020). Cloud of things: Architecture, applications and challenges. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-020-02448-3
[31] Rahman, A., Dash, S., Ahmad, M., Iqbal, T. (2021). Mobile cloud computing: A green perspective. Intelligent Systems, Lecture Notes in Networks and Systems Book Series (LNNS, volume 185), 523-533.
[32] Rahman, A., Abbas, S., Gollapalli, M., Ahmed, R., Aftab, S., Ahmad, M., Khan, M.A., Mosavi, A. (2022). Rainfall prediction system using machine learning fusion for smart cities. Sensors, 22(9): 1-15. https://doi.org/10.3390/s22093504
[33] Jamal, M., Zafar, N.A., Rahman, A., Musleh, D., Gollapalli, M., Chabani, S. (2022). Modeling and verification of aircraft takeoff through novel quantum nets. Computers, Materials and Continua, 72(2): 3331-3348. http://dx.doi.org/10.32604/cmc.2022.025205
[34] Ibrahim, N.M., Gabr, D.G.I., Rahman, A., Dash, S., Nayyar, A. (2022). A deep learning approach to intelligent fruit identification and family classification. Multimedia Tools and Applications, 81: 27783-27798. https://doi.org/10.1007/s11042-022-12942-9
[35] Gollapalli, M.A.S., Rahman, A., Musleh, D., Ibrahim, N.M., Khan, M.A., Abbas, S., Atta, A., Khan, M.A.A., Farooqui, M., Ahmed, M.I.B. (2022). A neuro-fuzzy approach to road traffic congestion prediction. Computers, Materials and Continua, 73(1): 295-310. http://dx.doi.org/10.32604/cmc.2022.027925
[36] Rahman, A., Alqahtani, A., Aldhafferi, N., Nasir, M.U., Khan, M.F., Khan, M.A., Mosavi, A. (2022). Histopathologic oral cancer prediction using oral squamous cell carcinoma biopsy empowered with transfer learning. Sensors, 22(10): 3833. https://doi.org/10.3390/s22103833
[37] Khan, M.A., Abbas, S., Atta, A., Ditta, A., Alquhayz, H., Khan, M.F., Rahman, A., Naqvi, R.A. (2020). Intelligent cloud based heart disease prediction system empowered with supervised machine learning. Computers, Materials & Continua, 65(1): 139-151. http://dx.doi.org/10.32604/cmc.2020.01141
[38] Ahmad, M., Qadir, M.A., Rahman, A., Zagrouba, R., Alhaidari, F., Ali, T., Zahid, F. (2020). Enhanced query processing over semantic cache for cloud based relational databases. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-020-01943-x
[39] Rahman, A., Dash, S., Luhach, A.K., Chilamkurti, N., Baek, S., Nam, Y. (2019). A Neuro-fuzzy approach for user behavior classification and prediction. Journal of Cloud Computing, 8(1): 1-15. https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-019-0144-9.
[40] Rahman, A., Sultan, K., Das, S., Khan, M.A. (2018). Management of resource usage in mobile cloud computing. International Journal of Pure and Applied Mathematics, 119(16): 255-261.
[41] Rahman, A., Sultan, K., Naseer, I., Majeed, R., Musleh, D., Gollapalli, M.A.S., Chabani, S., Ibrahim, N., Siddiqui, S.Y., Khan, M.A. (2021). Supervised machine learning-based prediction of COVID-19. Computers, Materials & Continua, 69(1): 21-34.

https://doi.org/10.32604/cmc.2021.013453
[42] Alotaibi, S.M., Rahman, A., Basheer, M.I., Khan, M.A. (2021). Ensemble machine learning based identification of pediatric epilepsy. Computers, Materials & Continua, 68(1): 149-165. http://dx.doi.org/10.32604/cmc.2021.015976
[43] Alhaidari, F., Almotiri, S.H., Ghamdi, M.A., Khan, M. A. Rehman, A., Abbas, S., Khan, K.M., Rahman, A. (2021). Intelligent software-defined network for cognitive routing optimization using deep extreme learning machine approach. Computers, Materials & Continua, 67(1): 1269-1285. http://dx.doi.org/10.32604/cmc.2021.013303
[44] Rahman, A., Mahmud, M., Iqbal, T., Saraireh, L., Kholidy, H.A., Gollapalli, M.A.S., Musleh, D., Alhaidari, F., Almoqbil, D., Ahmed, M.I.B. (2022). Network anomaly detection in 5G networks. Mathematical Modelling of Engineering Problems, 9(2): 397-404. https://doi.org/10.18280/mmep.090213
[45] Zaman, G., Mahdin, H., Hussain, K., Rahman, A., Abawajy, J., Mostafa, S.A. (2021). An ontological framework for information etraction from diverse scientific sources. IEEE Access, 9: 42111-42124. https://doi.org/10.1109/ACCESS.2021.3063181
[46] Rahman, A., Dash, S., Luhach, A.K. (2021). Dynamic MODCOD and power allocation in DVB-S2: A hybrid intelligent approach. Telecommun Syst., 76(1): 49-61. https://doi.org/10.1007/s11235-020-00700-x
[47] Rahman, A., Asif, R.N., Sultan, K., Alsaif, S.A., Abbas, S., Khan, M.A., Mosavi, A. (2022). ECG classification for detecting ECG arrhythmia empowered with deep learning approaches. Computational Intelligence and Neuroscience, 2022: 6852845. https://doi.org/10.1155/2022/6852845
[48] Rahman, A., Nasir, M.U., Gollapalli, M., Alsaif, S.A., Almadhor, A.S., Mehmood, S., Khan, M.A., Mosavi, A. (2022). IoMT-based mitochondrial and multifactorial genetic inheritance disorder prediction using machine learning. Computational Intelligence and Neuroscience, 2022: 2650742. https://doi.org/10.1155/2022/2650742
[49] Rahman, A., Nasir, M.U., Gollapalli, M., Zubair, M., Saleem, M.A., Mehmood, S., Khan, M.A., Mosavi, A. (2022). Advance genome disorder prediction model empowered with deep learning. In IEEE Access, 10: 70317-70328. https://doi.org/10.1109/ACCESS.2022.3186998
[50] Arooj, S., Rahman, A., Zubair, M., Khan, M.F., Alissa, K., Khan, M.A., Mosavi, A. (2022). Breast cancer detection and classification empowered with transfer learning. Front Public Health. https://doi.org/10.3389/fpubh.2022.924432
[51] Nasir, M.U., Ghazal, T.M., Khan, M.A., Zubair, M., Rahman, R., Ahmed, R., Hamadi, H.A., Yeun, C.Y. (2022). Breast cancer prediction empowered with fine-tuning. Computational Intelligence and Neuroscience, 2022: 5918686. https://doi.org/10.1155/2022/5918686
[52] Rahman, A., Ahmed, M., Zaman, G., Iqbal, T., Khan, M.A.A., Farooqui, M., Ahmed, M.I.B., Ahmed, M.S., Nabeel, M., Omar, A. (2022). Geo-spatial disease clustering for public health decision making. Informatica, 46(6): 21-31. http://dx.doi.org/10.31449/inf.v46i6.3827
[53] Asif, R.N., Abbas, S., Khan, M.A., Rahman, A., Sultan, K., Mahmud, M., Mosavi, A. (2022) Development and validation of embedded device for electrocardiogram arrhythmia empowered with transfer learning. Computational Intelligence and Neuroscience, 2022: 5054641. https://doi.org/10.1155/2022/5054641
[54] Nasir, M.U., Zubair, M., Ghazal, T.M., Khan, M.F., Ahmad, M., Rahman, A., Hamadi, H.A., Khan, M.A., Mansoor, W. (2022) Kidney cancer prediction empowered with blockchain security using transfer learning. Sensors, 22: 7483. https://doi.org/10.3390/s22197483
[55] Rahman, A., Qureshi, I.M., Malik, A.N., Naseem, M.T. (2016). QoS and rate enhancement in DVB-S2 using fuzzy rule based system. Journal of Intelligent & Fuzzy Systems, 30(1): 801-810. http://dx.doi.org/10.3233/IFS-151802
[56] Rahman, A., Qureshi, I.M., Malik, A.N., Naseem, M.T. (2016). Dynamic resource allocation in OFDM systems using differential evolution and Fuzzy Rule Base System. Journal of Intelligent & Fuzzy Systems, 26(4): 2035-2046. http://dx.doi.org/10.3233/IFS-130880
[57] Rahman, A., Alhaidari, F. (2019). The Digital library and the archiving system for educational Institutes. Pakistan Journal of Information Management & Libraries 20(1): 94-117. https://doi.org/10.47657/2018201453
[58] Rahman, A. (2019). Optimum information embedding in digital watermarking. Journal of Intelligent & Fuzzy Systems, 37(1): 553-564. http://dx.doi.org/10.3233/JIFS-162405
[59] Rahman, A. (2019). Memetic computing based numerical solution to Troesch problem. Journal of Intelligent & Fuzzy Systems, 37(1): 1545-1554. http://dx.doi.org/10.3233/JIFS-18579
[60] Rahman, A., Alhaidari, F., Musleh, D., Mahmud, M., Khan, M.A. (2019). Synchronization of virtual databases: A case of smartphone contacts. Journal of Computational and Theoretical Nanoscience, 16(4): 1740-1757. http://dx.doi.org/10.1166/jctn.2019.8115