Al-Rafidain Journal of Computer Sciences and Mathematics (RJCM)

www.csmj.mosuljournals.com

Real Time System Scheduling Approach:Survey

Dhuha B. Abdullah [1],* Israa Nasir Abood [2] Raya akram hamdi [3]

Department of Computer Science, College of computer science and mathematics, Mosul University, Mosul, Iraq [1.2.3]
**Corresponding author. Email: prof.dhuha_basheer@uomosul.edu.iq [1]*

Article information

Article history:
Received : 12 /6/2022
Accepted : 5/12/2022
Available online :

Abstract

Real-time systems play a major role in our today's life. They are used in essential control systems that rely on timely response and determined outcomes to work. The main real-time scheduling algorithms for both soft and hard real-time system is presented in this paper, both in processors uniprocessor and multiprocessor schemes. The effectiveness of scheduling is derived from various factors, including hardware configuration, real-time application type, and real-time problem complexity. This review presents a characterization of scheduling techniques to help the researcher to get enough knowledge in real-time systems with adequate scheduling schemes to reach their timeliness characteristic. In this paper, we aim to investigate the scheduling attributes and scope of research in real-time computing, and classify real-time system in two categories: algorithms for multiprocessor scheduling and algorithms for uniprocessor scheduling. Furthermore, gives special attention to characteristic and their task, that is one of the following original contributions to real-time scheduling algorithms.

Keywords:
Scheduling Algorithms, Real-time Systems, Hard RTS, Soft RTS schedulable, Priority driven Algorithms; EDF (Earliest Deadline First); RM (Rate Monotonic)

1. NTRODUCTION TO REAL TIME

Real-time systems are computing systems that will always perform their operations within specified strict deadlines [1,2]. In a Real Time, system, processes relate to tasks that have defined features such as a deadline, execution time, and release time. Because the real-time system has deadline constraints, a real-time task can be classed into one of three categories: hard, soft, or firm, based on the consequences of missing the deadline. If a deadline is missed on a hard real-time task, the system will fail completely and catastrophic consequences will result.

A deadline can be missed in a soft real-time system, but it will not result in a total failure, simply a reduction in system stability. A real-time task is firm since if the deadline is missed, the produced outcome will be useless and cause no damage to the system [3].

A large number of different applications use hard real-time systems today, including automotive electronics, avionics, space systems, medical systems, household automation, and robotics [4]. To get the desired and predictable behavior from the system, real-time scheduling algorithms are of great importance. In addition, the scheduling of tasks and the management of processor resources on different processors are equally important issues when designing algorithms for scheduling in multiprocessor systems.

Scheduling tasks in a global or hierarchical manner serves as the manager, is one of the generic methods. It then According to the general principle, divide the tasks between the processor's characteristics. Then, each processor can independently schedule based on its own approaches. There are many types of algorithms, and choosing an appropriate one has a significant impact on the behavior of the real-time system. Also, other critical Scheduling methods based on real-time criteria, including response time and average schedule rate, energy consumption, and fault tolerance are also important considerations in real-time scheduling methods [2].

The use of multi-core processors has increased as the number of applications of embedded real-time systems in modern life has grown, as has the necessity for greater processing. By

increasing the processing power, energy consumption will increase.

Consequently, effective energy use will become increasingly important. There are a number of tasks that must be performed, each with different requirements. task, how distinct tasks are scheduled on processors is of greater importance. Gradually, real-time applications have in this field, this role has taken on a major role. Therefore, scheduling plays an important role. algorithms should simultaneously handle should take into account both job time limitations and energy consumption.

According to the literature, a variety of in addition to the various scheduling algorithms, several other factors, such as priority assignment strategies (static or dynamic), task models, etc., were also discussed. (Periodic or sporadic), energy optimization, and more [4]. Scheduling algorithms for automotive applications are presented in this paper, describing scheduling algorithms, this paper recommends a taxonomy of scheduling algorithms that can help automotive designers decide on the appropriate scheduling algorithm [5]. A taxonomy of preemptive real-time tasks is described multiprocessors; however, they did not specifically address automotive applications.

This work is organized as follows. In Section II, we review related literature. In section III, we discuss scheduling algorithms for uniprocessor systems, such as the RM and EDF algorithms. A comparison between uniprocessor and multiprocessor scheduling algorithms is presented in Section IV. The conclusion and recommendations of this work are presented in section V.

2. LITERATURE REVIEW

A summary of the work of different researchers on real-time processor schedulers from 2019 to 2021 is provided in the following table.

Based on the work of Beni et al. [6], a hierarchical schedule was developed for Linux that is optimized for virtualizing containers and is compatible with LXC containers with multiple CPUs. We provide a scheduler that is constructed by changing the real-time control group's mechanism so that its real-time run queues can be scheduled by the SCHED DEADLINE policy. Researchers have shown that using the new scheduler, real-time applications can be scheduled inside LXC containers as predicted by previous theoretical CSF analyses. The scheduler is easier to set up and produces superior results, while also using less real-time computing capacity inside the system, so it can be used for any real-time application reduction runs.

In their [7] paper, Baital and Chakrabarti proposed a scheduling algorithm where random tasks of varying periodicity and duration are generated at different intervals. Especially in battery-powered systems, energy consumption is a critical design consideration. As part of our work on schedule, we have extended our efforts to heterogeneous multicore systems (HMS) architectures, where real-time tasks are distributed accordingly while still meeting deadlines. Using heterogeneous multicore schedulers, the researchers demonstrated that commercially available heterogeneous multicore processors can be made. By comparing the performance of our model to some popular and novel scheduling techniques, the authors found our model performs exceptionally well in all cases and reduces energy consumption by a significant amount.

As part of [8], Pradhan suggests It consists of a RR algorithm modified so that the initial quantum is equal to the burst time for the first task. Whenever a new task is added to the ready queue, the quantum of the ready queue The time is now calculated by averaging the burst times of all tasks in the queue, including the new one. Turnaround time and wait times were significantly improved in the MATLAB experiments.

The proposed algorithm by I. S. Rajput in [9] addressed the shortcomings of the RR algorithm, which does not take priority into account. The process will be arranged in the queue according to their priority first. This algorithm will then be used in the first round in order to perform the processes. The second round will reorder the tasks based on the remaining burst time. According to MATLAB evaluation results, this proposed algorithm has an advantage over the Average waiting time, average turnaround time, and context switching of the traditional RR algorithm.

D. Khokhar and A. Kaushik published an algorithm in [10] that resolves the problems with traditional RR algorithms. To determine the best quantum time, they developed a new approach based on the mean and mean of the tasks' burst times. Based on these experiences, a new algorithm has proven to be superior in terms of reducing waiting time and turnaround time.

In 2020, Sanj M S et al. [19] To increase efficiency without affecting classical Round Robin (RR) functionality, ERR was suggested. This algorithm is applied and tested by the CloudSim toolkit. In contrast to classical RR, ERR minimizes the waiting time for a specific number of tasks in a specific number of cloudlets.

In 2020, Yong Shi and et al. [20] In order to improve Min-min's efficiency, the BMin algorithm is proposed. Based on the results of a cloudsim simulation, the proposed algorithm reduces completion time, maximizes throughput, and enhances resource load balance.

3. Real time system classification

Real-time computing is the concept of computing in real-time. In order for temporary correctness to be ensured, tasks must be completed before the deadline. Failure to meet the deadline will have different consequences depending on the system [21, 22]. Figure 1 shows the classification

- Soft real-time:

In situations in which a deadline is missed, the quality of the result of a task decline. Eventually, the performance of the system will decline when too many deadlines are missed. When deadlines are missed, the system does not fail. An example of soft real-time is the capture of temperature, wind, humidity, and other sensors at the weather station.

- Firm real-time:

When a deadline is not met, the quality of the outcome of a task drops to zero. This does not mean that the system will collapse. The system degrades when too many deadlines are missed, as it does with the soft deadline. Streaming video is an example of such a system.

- Hard real-time:

In real-time deadline situations, the quality of the results is zero when deadlines are missed. More importantly, if one of the deadlines is missed, the system will fail completely and even result in catastrophic consequences. Hard real-time systems demand that tasks meet their deadlines at any system load by all means possible. The ABS system of a car can be viewed as an example of a real-time system with a hard deadline.

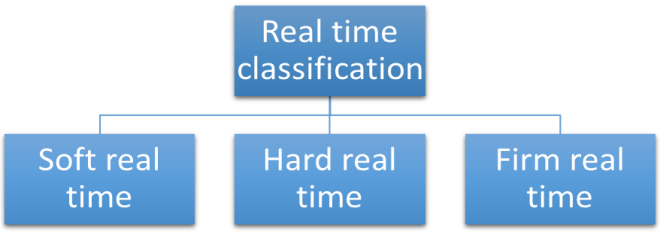

Figure 1: Real Time system classification

Table 1. Scheduling algorithms for real-time: summary of literature review

Ref. and Year	*Real Time Classification*	*Significant Results*	*Future Work Suggestions*
[6] 2019	Soft RT	Current CSF analyses indicate that real-time LXC-based scheduling performed using the new scheduler behaves as expected.	It is our intention to experiment with the suggested scheduler in a multi-CPU container-based simultaneous real-time activity setting in the near future.
[7] 2019	Hard RT	Compared to some popular and novel schedules, our approach works very well in all cases, and uses far less energy.	Implementing dependent task sets will be easier with this feature.
[8] 2019	Hard RT	By replicating random program interactions, the authors demonstrate the utility of Odyn on a test bench. In several scenarios, they show how stalemate avoidance works.	Research project ideas include calculating the appropriate memory allocation for PEs based on their performance requirements.
[9] 2020	Soft RT	As a result of the testing, the suggested algorithm shows 28.73% more completion time reduction and 46.31% greater resource usage than the other alternatives.	Thus, fault-tolerant scheduling of huge data is an important approach to consider
[10] 2020	Hard RT	The scheduling success rate of this method is much higher than the original approach.	The algorithm will then switch between tasks and context.
[11] 2020	Soft RT	Even with limited energy storage and flight duration, the DRL method can be used for real-time applications.	Using our approach, we will simultaneously optimize power allocation, trajectory planning, and numerous UAV scenarios to deal with more complex challenges
[12] 2020	Soft RT	The model was shown to perform well by using created task sets efficiently and using significantly less energy than a few others widely used and novel scheduling methods.	Validating the scheduling model will be our next project. We will ensure reliability and fault tolerance.
[13] 2020	Hard RT	In this article, we examine the schedulability of transiently supplied CPUs with NVMs in real time. With the RTC, it is possible to analyze in real-time the execution time of multi-tasks based on the quantitative behavior of the energy harvester over time.	A future study objective would be to minimize the threshold voltage search overhead without approximation
[14] 2021	Hard RT	Increasing the ability to predict the behavior of Earliest First Deadline (EDF) algorithms to reduce the number of relay tasks.	A real-time system utilizing the earliest deadline first (EFDF) scheduling algorithm reduced the complexity time of older tasks.
[15] 2021	Soft RT	The article discusses how the quality of the real-time scheduling algorithm affects the throughput capacity and response time of the real-time system	A single processor scheduling system, a multiprocessor scheduling system, a distributed scheduling system, and a single processor scheduling algorithm RMS, EDF, and LLF can be implemented depending on the system environment

4. REAL TIME SCHEDULING ALGORITHM

The functionality of a real-time system is described by numerous computational and logical activities. As these activities can't be performed in a random order, a precedence graph highlights their interconnectedness. In general, scheduling algorithms are used to determine the best way to distribute CPUs and networks are shared resources for computing activities. Various algorithms are offered in scheduling literature based on the restrictions enforced. Among these algorithms are periodic, independent tasks on uniprocessors and Distributed tasks with periodicity and aperiodicity [23,24].

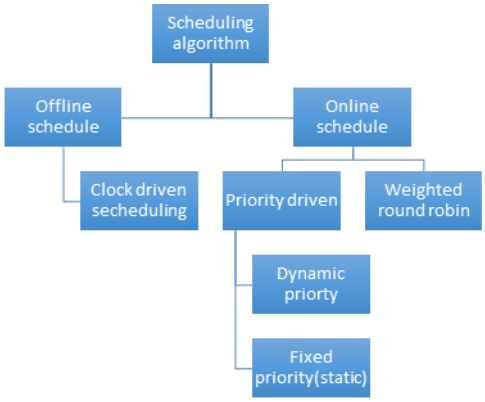

Figure 2: REAL TIME SCHEDULING CLASSIFICATION

Offline scheduling:

All the scheduling decisions for the real-time system are stored in a scheduling table, along with the activation times for all tasks. Scheduling decisions are therefore made before a process has started and are based on an understanding of how it will behave [23].

Clock driven scheduling

Tasks can be triggered by clock-driven approaches if their release times are known and all tasks are periodic. An offline schedule can then be created based on this information. Task dependency is a constraint on the scheduler, which must be met. A schedule is built for a hyper period if all tasks are released synchronously. We then arrive at a periodic schedule, or cyclic schedule, that concludes this process. Generally, when the tasks are independent, the schedules can be worked out by an algorithm, including priority-driven scheduling algorithms. The scheduling problem becomes NP-complete if any precedence or exclusion constraints exist[22].

Online scheduling:

The scheduling algorithm makes decisions Priorities are assigned during runtime according to certain rules. Schedules are typically stored as tables. a separate database, i.e., in a static system, but it can be used in Systems with dynamic arrivals and departures of jobs (e.g., embedded systems) [24].

Weighted round robin

In a round-robin schedule, tasks are assigned a time slice and scheduled in order. It goes back to the queue and waits for the next round after it has consumed its time slice or been yielded to another task. Modern operating systems are based on this scheduling algorithm. By categorizing processes into different priority levels, the Linux kernel implements completely fair queueing, which is a variation of round-robin. Typically, one round of scheduling is performed at each level. There is the same problem with this algorithm as with the FIFO algorithm, which does not consider the deadlines for tasks [25].

Priority driven

The Tasks may be assigned fixed or dynamic properties by the scheduler in a priority-driven scheduling approach. During compilation, this assignment is applied to each task in the task set. A queue of ready tasks is maintained by the online scheduler, which Organizes the tasks by priority and executes them accordingly. Schedules that are prioritized based on fixed priorities or dynamic priorities are classified as priority-driven scheduling. Below, two exemplary algorithms, one for each class, are briefly described for each domain [26].

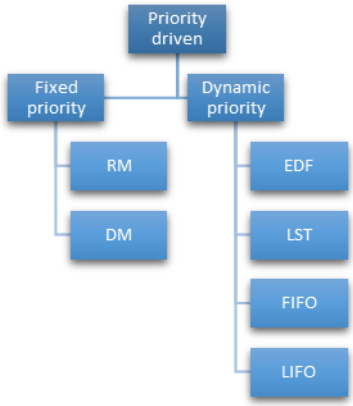

Figure 3: Ppriority driven classification

Static Algorithms

In this section, we describe Static-priority scheduling algorithms are some of the most important:

- RM (Rate Monotonic)

For uniprocessor systems, an algorithm for determining priority scheduling is RM, which is static and preemptive. Due to the demand rate being higher, shorter period tasks are more likely to be executed. If the period is shorter, the priority would increase as well. Therefore, periodic tasks are used. A number of assumptions have been incorporated into the RM algorithm [25]:

1) Periodic tasks are ready to begin executing during the beginning of each period T and have a fixed runtime.
2) In other words, the Tasks (D) have an implied deadline (D) period end, since D=T.
3) The tasks are independent of one another and do not interfere with one another.
4) As a result of time spent on context switching and exchanging, no scheduling overhead is assumed.

- DM (Deadline Monotonic)

The D*T (constrained deadline) scheduling algorithm is similar to RM and differs from RM in that it has a fixed date. The task priority in RM can be determined by the deadline, so it is treated as a special case of DM. Consequently, shorter deadlines are assigned a higher priority.

RM has undergone some improvements to enhance its performance. So, when the task is shared with their resources, we can also make use of the RM. Using a

technique called semaphore, we can prevent the simultaneous use of shared resources. The task will lock when it reaches the critical section, and once it exits it is released [24,25].

Dynamic Algorithms

Dynamic-priority scheduling algorithms include the following [25,26]:

- EDF (Earliest Deadline First)

The EDF algorithm determines priority via dynamic scheduling priorities based on deadlines. So, the highest priority would be given to tasks that are approaching their deadline. In addition, in addition, it has the capability of maximizing the efficiency of the system because of its preemptive nature. In addition to RM, EDF also holds the same conditions and assumptions.[25]

- LLF (least Laxity First)

Accordingly, Since the LLF algorithm prioritizes according to laxity, it will have a higher priority if it had less laxity. A laxity interval accounts for the period during which a task is relaxed. be executed.

Therefore, Laxity time determines if two tasks are the same automatically stop another execution when they have the same laxity time. As a result, there will be frequent context switches. Nevertheless, ignoring A dynamic scheduling algorithm optimized for LLF's derived cost, similar to EDF [26].

5. EXAMPLES OF PRACTICAL REAL-TIME APPLICATIONS:

In recent years, real-time has increasingly been used to describe many computing systems and applications that are related to time, including time trackers, gaming systems, and information services [21, 22].

- General control management systems

such as the ones found in avionic systems. Realtime engine controllers are responsible for automatic navigation and detection of hardware malfunctions or damages through reading sensors and processing their parameters and reacting within an acceptable delay. Another critical application is the air traffic control system [23].

- Mobile and communication systems.

Message passing and delay guarantees are required by wireless communication systems, for example, for Applications that have a large number of nodes, such as automotive and industrial applications.[20].

- Data distribution systems

Users are notified of important information in a short period of time (a few minutes or less). A system like this is used to notify passengers about accidents, schedule delays, and other changes in transport systems [21].

- General-purpose computing

This is common in financial and banking systems.

- Multimedia and entertainment systems:

Streaming audio and video are forms of multimedia information. In information processing, communication between multimedia servers and receivers can be of great importance in terms of real-time requirements [21].

- Medical systems

There are peacemakers and medical monitors of treatments and surgery, for example.

- Industrial automation systems

They are used Controlling and monitoring production in factories processes. Sensors, for instance, continuously gather Measurements are sent to real-time controllers, which evaluate them and make adjustments when necessary needed. Systems such as these can be used for noncritical activities, such as logging and surveillance [22].

6. SCHEDULING ALGORITHMS FOR UNIPROCESSOR AND MULTIPROCESSOR SYSTEMS

Multiple scheduling algorithms are used in real-time systems with a single processor. There are two types of algorithms: static algorithms and priority-driven algorithms. A static algorithm is one that divides the processor time evenly among many tasks, such as Round Robin (RR). The focus of this section is on priority-driven algorithms. [26]

A priority-driven scheduling algorithm can either be fixed or dynamic. The priority assignment determines whether the priority is static or changes during running. This section The EDF and RM algorithms are the most commonly used priority-driven algorithms in real-time systems. [28,29]

An alternative name for the RM Scheduling Algorithm is Rate Monotonic Algorithm. A fixed or static priority scheduling algorithm is the RM algorithm. RM prioritizes tasks according to their period. This algorithm has the disadvantage that it does not provide a perfect result in low-load situations. In overloaded situations, RM performs better than dynamic scheduling. The RM algorithm gives the most chances of executing in the shortest period of time. [27]

Earlier deadline first scheduling algorithms are also known as nearest deadline first scheduling algorithms [27]. Dynamic scheduling algorithms are based on the EDF algorithm. As soon as possible, the task must be completed. The earliest deadline determines the priority of a task. A task is utilized 100 percent when EDF Scheduling is used, regardless of whether it is loaded or not loaded equal to 1. When tasks are overloaded or cross-loaded, the processor's

utilization decreases exponentially [28]. As shown in Table 2, RM and EDF have their advantages and disadvantages.

TABLE 2. ADVANTAGES AND DISADVANTAGES OF RM AND EDF

Algorithms	Advantages	Disadvantages
Rate Monotonic	• It is easy to implement. • This is a common algorithm.	Utilization of CPU is wasted
Earliest Deadline First	Utilization of all processes	• Implantation difficulty • Overloaded conditions can lead to misbehavior

RM and EDF behave differently on the same task set as shown in Figure 3. Suppose there are three tasks in a task set, Ti, Ci, and Pi, where Ti represents the computation time and period of each task. A task T1(2,6), a task T2(3,8), and a task T3(2,12) are included. The priority in RM is determined by the period, as shown in Figure 3(a). Thus, the most important task is the one with the shortest period. The priority of the EDF changes based on the deadline in Figure 3(b). As a result, the task with the shortest deadlines at each time interval is given the highest priority.

Figure 4 This case study illustrates how RM misbehaves under certain conditions. Assume you have two tasks T1(2,5) and T2(4,7) in a task set. Due to T1's higher priority than T2, T1 will preempt every instance of T2, and sometimes it can result in a deadline being missed. Figure 4(b) illustrates how the EDF can schedule this task set without missing any deadlines. In [29], it states that "For larger task sets, the number of preemptions caused by RM increases, thus the overhead due to context switching is higher under RM".

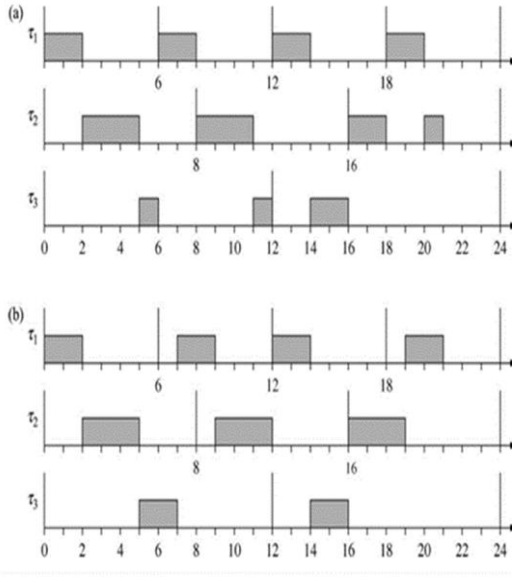

Figure 3. (a) RM and (b) EDF scheduling comparison [28]

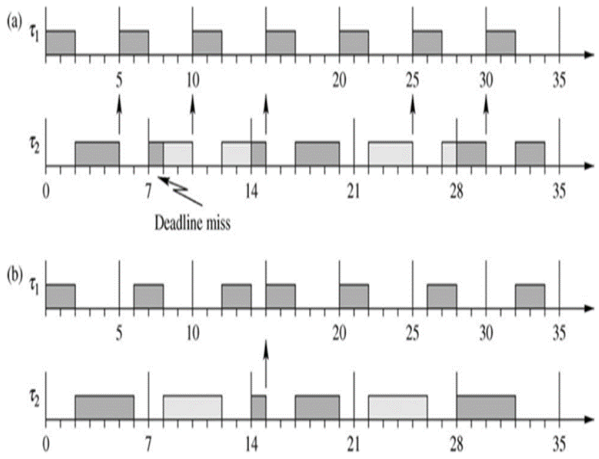

Figure 4. (a) RM and (b) EDF scheduling comparison [28]

Increasingly complex and heavy computations require more than one processor as time goes on. The scheduling scheme for multiprocessor systems differs from that for uniprocessor systems. To find the best scheduling algorithm, many research works have been conducted in this field. Fig. Multiprocessor systems are classified according to the algorithm in figure 5. They can be divided into classic algorithms, heuristic algorithms, and evolutionary algorithms. While most algorithms in the classic category are not specifically designed for multiprocessor environments, they do achieve a lesser time complexity when used in multiprocessor systems. Classic algorithms don't guarantee an optimal solution, which is one of their major drawbacks. As well as heuristics and evolutionary algorithms, which achieve a near-optimal result at the expense of more running time, there are also heuristic algorithms [30].

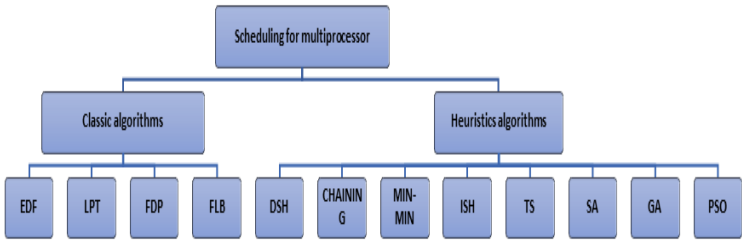

Figure 5. Scheduling algorithms for multiprocessor systems [29,30]

In Table 3, authors in [30,32] This paper compares some uniprocessor and multiprocessor scheduling algorithms from a variety of metrics, including priority, CPU utilization, context switching, optimality, likelihood of missing deadlines, response time, predictability, effectiveness, and limitations. Compared to other uniprocessor algorithms, the Instantaneous Utilization Factor (IUF) scheduling algorithm performs better than that of the Modified Instantaneous Utilization Factor (MIUF). utilization factor In comparison to other multiprocessor algorithms, MIUF provides faster response times, higher CPU utilization, and easier context switching [31][32].

TABLE 3. Comparison between Uniprocessor and Multiprocessor Algorithms [30,31,32]

Algorithms / Performance Metric	Uniprocessor Algorithms				Multiprocessor Algorithms			
	EDF	**LLF**	**MUF**	**IUF**	**EDZL**	**ILLF**	**MMUF**	**MIUF**
Priority	D	D	Hybrid	D	D	D	Hybrid	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 cosideration	Work in transient overload	maximize utilization bound of schedule	Context switching overhead is low	Less context switching	Optimal for noncritical tasks	Improves context switching, response time and CPU utilization
Limitations	Not Work in overload, not optimal for pro.>1	In laxity tie, more context switches occurs	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	

7. CHALLENGES AND ISSUES

As multiprocessor systems become more prevalent in today's computing environment, they can be used as powerful computing solutions in hard real-time systems with the highest efficiency and reliability. One of the challenges in the field of computer engineering is scheduling multiprocessor systems. Real-time task scheduling in multiprocessor systems consists of determining which task from a set of tasks should be executed on which processor. The following issues may also arise [31,32]:

- Processor usage restrictions
- effective scheduleability tests
- Taking overhead (cost) into account
- Multiprocessor systems with limited task models
- Limitations on the use of shared resources (Resource Allocation)

8. CONCLUSION

A review of real-time scheduling algorithms is presented in this paper. In the field of scheduling algorithms in real-time systems, earlier studies have been reviewed and discussed. Also discussed in this paper are the most commonly used uniprocessor algorithms, which are RM and EDF. Additionally, multiprocessor scheduling algorithms have been described.

This paper discusses scheduling and hard RTS scheduling approaches. Tables have been compiled and compared to summarize the results. Furthermore, several common scheduling algorithms are examined. This is the best way to guarantee that tasks are completed on time under these conditions. Then, in the following part, in some hard RTS applications, there are several scheduling approaches will be discussed

As a result, the choice of scheduling algorithm depends on many factors, and no one algorithm is optimal for all systems due to the differences in their structure and needs. Research and development in evolutionary algorithms in scheduling can be done in the future.

9. REFERENCES

[1] Buttazzo, G.C.: 'Hard real-time computing systems: predictable scheduling algorithms and applications' (Springer Science & Business Media, 2011. 2011)

[2] Davis, R.I.: 'A review of fixed priority and EDF scheduling for hard real-time uniprocessor systems', ACM SIGBED Review, 2014, 11, (1), pp. 8-19

[3] Lindh, F., Otnes, T., and Wennerström, J.: 'Scheduling algorithms for real-time systems', Department of Computer Engineering, Malardalens University, Sweden, 2010

[4] Davis, R.I., and Burns, A.: 'A survey of hard real-time scheduling for multiprocessor systems', ACM computing surveys (CSUR), 2011, 43, (4), pp. 1-44

[5] Carpenter, J., Funk, S., Holman, P., Srinivasan, A., Anderson, J.H., and Baruah, S.K.: 'A Categorization of Real-Time Multiprocessor Scheduling Problems and Algorithms', in Editor (Ed.)^(Eds.): 'Book A Categorization of Real-Time Multiprocessor Scheduling Problems and Algorithms' (2004, edn.), pp.

[6] Abeni, L., Balsini, A., and Cucinotta, T.: 'Container-based real-time scheduling in the linux kernel', ACM SIGBED Review, 2019, 16, (3), pp. 33-38

[7] Baital, K., and Chakrabarti, A.: 'Various approaches for high throughput and energy efficient scheduling of real-time tasks in multicore systems', in Editor (Ed.)^(Eds.): 'Book Various approaches for high throughput and energy efficient scheduling of real-time tasks in multicore systems' (IEEE, 2019, edn.), pp. 402-405

[8] Dauphin, B., Pacalet, R., Enrici, A., and Apvrille, L.: 'Odyn: Deadlock Prevention and Hybrid Scheduling Algorithm for Real-Time Dataflow Applications', in Editor (Ed.)^(Eds.): 'Book Odyn: Deadlock Prevention and Hybrid Scheduling Algorithm for Real-Time Dataflow Applications' (IEEE, 2019, edn.), pp. 88-95

[9] Chen, H., Wen, J., Pedrycz, W., and Wu, G.: 'Big data processing workflows oriented real-time scheduling algorithm using task-duplication in geo-distributed clouds', IEEE Transactions on Big Data, 2018, 6, (1), pp. 131-144

[10] Cao, S., and Bian, J.: 'Improved DAG Tasks Stretching Algorithm Based on Multi-core Processors', in Editor (Ed.)^(Eds.): 'Book Improved DAG Tasks Stretching Algorithm Based on Multi-core Processors' (IEEE, 2020, edn.), pp. 18-21

[11] Nguyen, K.K., Vien, N.A., Nguyen, L.D., Le, M.-T., Hanzo, L., and Duong, T.Q.: 'Real-time energy harvesting aided scheduling in UAV-assisted D2D networks relying on deep reinforcement learning', IEEE Access, 2020, 9, pp. 3638-3648

[12] Chen, J.-J., Shi, J., von der Brüggen, G., and Ueter, N.: 'Scheduling of real-time tasks with multiple critical sections in multiprocessor systems', IEEE Transactions on Computers, 2020, 71, (1), pp. 146-160

[13] Lee, D., Jung, H., and Yang, H.: 'Real-time schedulability analysis and enhancement of transiently powered processors with NVMs', IEEE Transactions on Computers, 2020, 70, (3), pp. 372-383

[14] Pradhan, P., Behera, P.K., and Ray, B.: 'Modified round robin algorithm for resource allocation in cloud computing', Procedia Computer Science, 2016, 85, pp. 878-890

[15] Rajput, I.S., and Gupta, D.: 'A priority based round robin CPU scheduling algorithm for real time systems', International Journal of Innovations in Engineering and Technology, 2012, 1, (3), pp. 1-11

[16] Balharith, T., and Alhaidari, F.: 'Round robin scheduling algorithm in CPU and cloud computing: a review', in Editor (Ed.)^(Eds.): 'Book Round robin scheduling algorithm in CPU and cloud computing: a review' (IEEE, 2019, edn.), pp. 1-7

[17] A. Rahman, S. Dash, M. Ahmad, T. Iqbal, "Mobile Cloud Computing: A Green Perspective," Intelligent Systems, Lecture Notes in Networks and Systems book series (LNNS, volume 185), pp. 523-533, 2021.

[18] Rahman, A. U., Dash, S., & Luhach, A. K. (2021). Dynamic MODCOD and power allocation in DVB-S2: a hybrid intelligent approach. Telecommunication Systems, 76(1), 49-61.

[19] Sanaj M S, J. P. P. M. (2020). An Enhanced Round Robin (ERR) algorithm for Effective and Efficient Task Scheduling in cloud environment. IEEE.

[20] Yong Shi, K. S., Steven Kemp and Jameson Hodge. (2020). A Task Scheduling Approach for Cloud Resource Management. Paper presented at the Fourth World Conference on Smart Trends in Systems,Security and Sustainability (WorldS4)

[21] Qamhieh, M.: 'Scheduling of parallel real-time DAG tasks on multiprocessor systems', Paris Est, 2015

[22] ALAHMAR, D.: 'Random Task Scheduler Algorithms as a Comparison and Access to the Best to Use in Real Time', International Journal of Scientific & Engineering Research, 2019, 1, (5), pp. 529-522.

[23] Voss, S.: 'Integrated task and message scheduling in time-triggered aeronautic systems', Duisburg-Essen University Munich, Germany, 2010

[24] Buttazzo, G.C.: 'Hard real-time computing systems: predictable scheduling algorithms and applications' (Springer Science & Business Media, 2011. 2011)

[25] Rouhifar, M., and Ravanmehr, R.: 'A survey on scheduling approaches for hard real-time systems', International Journal of Computer Applications, 2015, 131, (17), pp. 41-48

[26] D. G. Harkut, M. S. Ali, M. Poonam Lohiya, B. Principal, and nd yr, "Real-time Scheduler For Wireless Sensor Network : A Review." [Online]. Available: www.ijert.org

[27] H. Thakar, "Comparison between EDF_RM and EDF_DM in dynamic scheduling algorithm with sporadic task," vol. 1, 2016, [Online]. Available: www.ijsdr.org49

[28] G. C. Buttazzo, "Rate Monotonic vs. EDF: Judgment Day," 2005

[29] M. Rouhifar and R. Ravanmehr, "A Survey on Scheduling Approaches for Hard Real-Time Systems," International Journal of Computer Applications, vol. 131, no. 17, pp. 41–48, Dec. 2015, doi: 10.5120/ijca2015907656.

[30] M. Rouhifar and R. Ravanmehr, "A Survey on Scheduling Approaches for Hard Real-Time Systems," International Journal of Computer Applications, vol. 131, no. 17, pp. 41–48, Dec. 2015, doi: 10.5120/ijca2015907656.

[31] A. Rahman, I.M. Qureshi, A.N. Malik, M.T. Naseem, "Dynamic Resource allocation for OFDM Systems using DE and Fuzzy Rule Base System", Journal of Intelligent & Fuzzy Systems (JIFS), vol. 26 (4), pp. 2035-2046, 2014.

[32] A. Rahman, "Applications of Hybrid Intelligent Systems in Adaptiven Communications", Modeling, Analysis and Applications of NatureInspired Metaheuristic Algorithms, Edition: 1st, Chapter: 10, pp. 183-217, Publisher: IGI Global, 2017

[33] A. Rahman, S.A. Alrashed, A. Abraham, "User Behavior Classification and Prediction using FRBS and Linear Regression" Journal of Information Assurance and Security, vol. 12, no. 3, pp. 86-93, 2017.

نهج جدولة النظام في الوقت الحقيقي: الاستبيان

ضحى بشير عبدالله اسراء ناصر عبود رايا اكرم حمادي
قسم علوم الحاسوب، كلية علوم الحاسوب والرياضيات، جامعة الموصل، الموصل، العراق
prof.dhuha_basheer@uomosul.edu.iq

تاريخ الاستلام:12/6/2022 تاريخ القبول:5/12/2022

الملخص

تلعب أنظمة الوقت الفعلي دورًا رئيسيًا في حياتنا اليوم. يتم استخدامها في أنظمة التحكم الأساسية التي تعتمد على الاستجابة في الوقت المناسب والنتائج المحددة للعمل. يتم تقديم خوارزميات الجدولة الرئيسية في الوقت الحقيقي لكل من نظام الوقت الفعلي الناعم والصعب في هذه الورقة ، في كل من أنظمة المعالجات أحادية المعالج ومتعددة المعالجات. يتم اشتقاق فعالية الجدولة من عدة عوامل ، بما في ذلك تكوين الأجهزة ونوع التطبيق في الوقت الفعلي ومدى تعقيد المشكلة في الوقت الفعلي. تقدم هذه المراجعة توصيفًا لتقنيات الجدولة لمساعدة الباحث في الحصول على معرفة كافية في أنظمة الوقت الفعلي مع مخططات جدولة مناسبة للوصول إلى خصائصها في الوقت المناسب. في هذا البحث ، نهدف إلى التحقيق في سمات الجدولة ونطاق البحث في الحوسبة في الوقت الفعلي ، وتصنيف نظام الوقت الفعلي إلى فئتين: خوارزميات لجدولة متعددة المعالجات وخوارزميات لجدولة أحادية المعالجات. علاوة على ذلك ، يولي اهتمامًا خاصًا للسمات المميزة ومهمتها ، وهي إحدى المساهمات الأصلية التالية في خوارزميات الجدولة في الوقت الفعلي.

الكلمات المفتاحية : خوارزميات الجدولة ، أنظمة الوقت الفعلي ، RTS الصلبة ، إمكانية جدولة RTS الناعمة ، الخوارزميات المدفوعة بالأولوية ؛ EDF (أقرب موعد نهائي أولاً) ؛ RM (معدل رتيب)