Asian Journal of Research in Computer Science

11(4): 35-51, 2021; Article no.AJRCOS.73758
ISSN: 2581-8260

Scheduling Algorithms Implementation for Real Time Operating Systems: A Review

Gulistan Ahmead Ismael[1*], Azar Abid Salih[1], Adel AL-Zebari[1], Naaman Omar[1], Karwan Jameel Merceedi[1], Abdulraheem Jamil Ahmed[1], Nareen O. M. Salim[1], Sheren Sadiq Hasan[1], Shakir Fattah Kak[1], Ibrahim Mahmood Ibrahim[1] and Hajar Maseeh Yasin[1]

[1]Duhok Polytechnic University, Duhok, Kurdistan Region, Iraq.

Authors' contributions

This work was carried out in collaboration among all authors. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/AJRCOS/2021/v11i430269
Editor(s):
(1) Prof. M. A. Jayaram, Siddaganga Institute of Technology, India.
Reviewers:
(1) Neelesh Jain, Sagar Institute of Research & Technology, India.
(2) R. Jemima Priyadarsini, Bishop Heber College, India.
Complete Peer review History: https://www.sdiarticle4.com/review-history/73758

Review Article

Received 01 June 2021
Accepted 11 September 2021
Published 11 September 2021

ABSTRACT

The term "Real-Time Operating System (RTOS)" refers to systems wherein the time component is critical. For example, one or more of a computer's peripheral devices send a signal, and the computer must respond appropriately within a specified period of time. Examples include: the monitoring system in a hospital care unit, the autopilot in the aircraft, and the safety control system in the nuclear reactor. Scheduling is a method that ensures that jobs are performed at certain times. In the real-time systems, accuracy does not only rely on the outcomes of calculation, and also on the time it takes to provide the results. It must be completed within the specified time frame. The scheduling strategy is crucial in any real-time system, which is required to prevent overlapping execution in the system. The paper review classifies several previews works on many characteristics. Also, strategies utilized for scheduling in real time are examined and their features compared.

Keywords: Operating system OS; RTOS; real time; scheduling algorism.

**Corresponding author: E-mail: gulistan.ismael@dpu.edu.krd;*

1. INTRODUCTION

Real-Time Operating System (RTOS) is crucial for mechanical and electrical systems that need real-time operation, since these systems would be unable to run properly without it [1]. A missed deadline may have severe repercussions in many real-time systems [2]. A number of industrial systems use the RTOS, including process control, avionics, and nuclear power plants [3]. Embedded software is used in the majority of actual operating systems, which are composed of hardware components that act as controllers for specific operations inside mechanical or electrical systems [4, 5]. The integrity of the results in Real - time data Systems is dependent on both the analytical conclusion of the computations and the time instant at which the conclusion is created [6, 7]. If at least one job in the system has defined timing restrictions, i.e., action must be done within a specified time period, a real time system is the name given to the system [8]. Meeting a deadline is the primary requirement of any real-time system, and term "deadline" refers to the point in time at which an action must be done. otherwise, the RTS is said to have failed [9]. Three distinct forms of Real Time-Systems exist: There are three types of real-time systems: hard, firm, and soft [10]. Hard Real Time System: If a deadline is missed, the whole company would collapse; some of the deadlines are vital for safety, a good example of this is an anti-missile system [11]. System in Soft Real Time The software does not fail if a deadline is missed; but the usefulness of the output decreases with time. Performance of the system will deteriorate. Every interactive application is an illustration of a soft real time system [12]. Firm Real Time System (RTS): this is a system that is almost identical to soft RTS. RTOS has been used in a broad variety of embedded devices for years in automation and computer science [13]. Such systems have been created to help regulate embedded algorithms on military equipment, systems and software for large switching systems [14]. The use of RTOS is generally performed on hard real-time systems [15, 16]. The RTOS has expanded dramatically in recent decades [17]. Applications of these systems are considered robust and always a designer challenge. These systems have to ensure that time constraints are met while executing complex activities [18]. A ROS is the sort of operating system intended to operate in real time. To manage programs capable of managing data collected without buffering, resulting in delays [19]. The technique for processing the time is measured in time increments that allow information to be shared [20, 21]. Time for processes the priority and coordination duties of event-based systems are changed. Interrupts when schedule systems switch jobs [22]. A number of commercial and non-commercial solutions are available [23]. Embedded systems operating systems for the globe today. Each one has its unique set of features, abilities and advantages. However, they all offer virtually the same service, in addition to downsides in general. The user should have the same ability in basic operations [24, 25].

The remainder of the essay is organized as follows: In the Section2, the background theory, in the Section 3 the related work of this survey is analyzed. In section 4 discussion, and finally the conclusion of this work presented in Section 5.

2. BACKGROUND AND THEORY

2.1 Concepts of Scheduling Algorithms

A RTO (Real Time Operating System) is an OS meant to serve in real time applications, often without buffer delays, which process data as it enters [26]. Time requirements for processing (including OS delays) are measured in 10 seconds or shorter increments [27]. A time-based, clearly defined, set timescales system are a reliable system [28]. Processing within the stated restrictions must be carried out or the system fails. Either you have an event or you share time [29]. Systems driven by events switch between activities on the basis of priority, whereas systems based on clock interruptions switches time-sharing systems. Most RTOSs utilize a preventive approach for scheduling [30, 31].

Software applications running on real-time operating systems offer important time requirements for real-time systems [32, 33]. These software tasks must be organized in accordance with software and hardware events. In real-time operating systems certain services for control of software tasks (priority-based pre-emption etc.) exist [34]. However, there is a need to plan algorithms in real-time systems in specific scenarios [35]. This is particularly necessary when time-critical software tasks need to be carried out throughout multiple working times and a certain schedule [36]. To fulfil the system requirements, different scheduling methods can be used [37]. For time critical jobs, the flight software running on the operating system in real time, particularly on satellite systems [38].

Embedded real-time systems are expanding progressively in order to run high-performance, multi-core architectural applications [39]. Efficient task plan models in these systems are extremely important to ensure that most of the jobs may be planned within their deadline to deliver the desired performance [40].

In the next Internet era, it is greatly wanted to reduce energy consumption of integrated computers [41]. One of the most efficient ways for both dynamic and static processor energy usage is Minimum Energy Point Tracking [42]. In the past several years, earlier studies suggested a number of MEPT techniques. Although edge computing applications generally demand low energy tasks with real-time guarantees. The problem is the time complexity in which to recognize MEPs and change voltages, which frequently prohibits the planning of tasks in real time [43].

Robot Operating System (ROS) supports insulating defects, quicker development and modularity, and the core reusability, thereby providing a broad and de facto standard for independent driving systems [44]. The GPUs also allow high-performance computation and are thus utilized for autonomous driving [45]. As real time processing requirements rise, techniques are being developed for meeting the real-time limit for ROS and GPUs [46, 47]. Regrettably, algorithms are not under investigation, which describe the ROS transport (publish/subscribe) model that may be restricted in execution, resulting to time waiting and a reduction of the reaction of the system as a whole [48]. In addition, ROS GPU workloads are also influenced by the ROS transport model since the time is taken to set up the central processing unit (CPU) [49].

Effective time operating systems (RTOS) should encourage the protocols of resource access to limit the maximum delay of priority inversions. Such protocols must be implemented lightly since their performance impacts planning [50, 51].

An essential challenge in soft re-timing system designs is the handling of algorithms that handle a large number of comparable data in a dynamic yet definite fashion utilizing a large number of tiny and diverse sizes of memory allocation/de-allocation [52, 53].

Real time systems are systems that rely on two elements. The first is the logical outcome of computing and the second is the moment when the results are produced [54, 55]. A process to finish its execution takes a certain period and the system will fail if the time is over for the process. Two real-time planning approaches exist: (1) dynamic static (2). The priorities remain the same for a job in the static algorithm and priorities are set at conception, with dynamic priority being assigned in due course [56].

The Realtime systems are in which the accuracy of the output depends not only on the logical conclusions of the calculations, but on the time of the output [57]. This indicates that the results must be generated within the specified time period [58]. The time limit for the system to reply is termed the time limit. Deadline fulfilment in any real-time system is an essential characteristic [59]. The ETS enables to complete applications in real time to fulfil their deadline utilizing the scheduling mechanism [60]. The planning approach is the core of any real-time system that makes decisions regarding the execution of tasks on the system in order to prevent overlap of any sort [61].

A major feature of the RTOS is the level of coherence about its length of time required for an application to be accepted and completed, the variability of which called 'jitter.' A 'hard' real-time OS (Hard RTOS) is less jittered than a 'soft' real-time OS (Soft RTOS) [62]. The late answer in a hard RTOS is incorrect, but a late response in a soft RTOS is okay [63]. The main objective for design is a soft or hard performance category rather than a high output [64]. An RTOS that typically or generally meet a time limit is a Soft Real-time OS, while it is a Difficult Real-time OS if it meets a time limit deterministically [65]. An RTOS has a sophisticated programming algorithm [66]. Scheduler flexibility allows the orchestration of process priorities by a larger, computer system, although a real-time OS is more often used to address a small number of applications [67]. Key elements in a real-time OS are minimum interrupt latency and minimum thread latency; with a real-time OS, it's more appreciated how quickly or reliably it reacts than how much work it can do in a particular time period [68].

2.2 Components of RTOS

A real-time operating system (RTOS), which controls system resources in a timely fashion and provides consistent foundation for the production of application code, is described as a program [69]. The RTOS must thus provide the temporal planning of processes, hardware control and access to RTOS resources. The RTOS kernel is accessed through an interface known as the System Call API [70]. This is a set of RTOS services functions [71]. Atomic processing must be performed, which implies the CPU cannot be paused while the kernel resources are changed. Implementation of a system call API function might be blocked or blocked [72]. The process does not halt a non-blocking system call, but a blocking call suspends the system until a requirement has been met. When a process requires Kernel services, it configures system call parameters and either conducts a particular trap command or sets a kernel event to force context into kernel space [68]. The kernel then checks the reasons for the caller process, carries out the requested action, and returns control and operational outcome to the user space process [73]. The following is a list of the major components of real-time operating system

- The Scheduler: This RTOS component specifies the sequence in which tasks may be done, which is often determined by their priority [74].
- Symmetric Multiprocessing (SMP): is a collection of the numerous distinct jobs that the RTOS may manage in order to do parallel processing [75].
- Function Library: It is a critical component of RTOS since it serves as an interface between the kernel and application code. This application allows you to communicate with the Kernel through a function library, guaranteeing that the application produces the required results [76].
- Memory Management: This component is essential to ensure that the system allocates memory to each program, which is the RTOS's most crucial function [77].
- Fast dispatch latency: It is the average time between a completion of the OS-identified job and the beginning of processing by the thread in the ready queue [78].
- User-defined data objects and classes: RTOS Systems make use of operating systems such as Windows or C++, which must be organized according to the purpose for which they are designed [79].

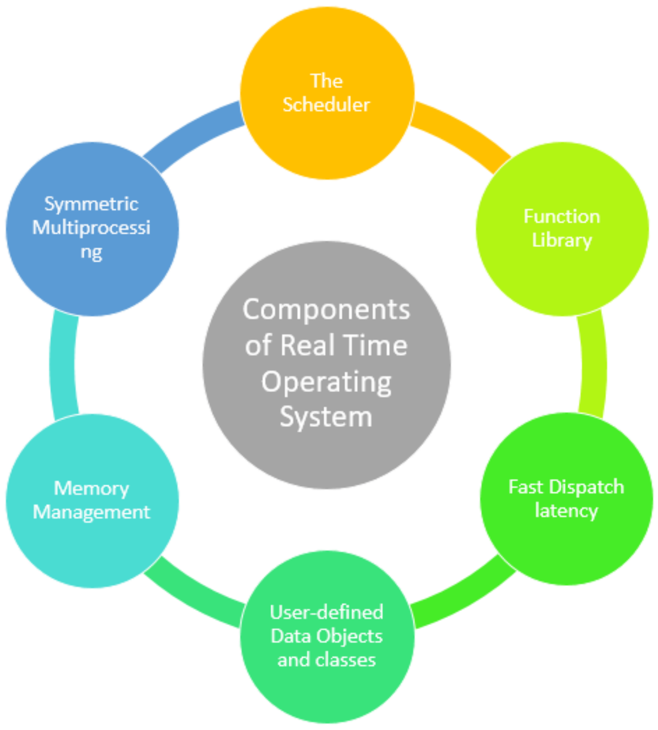
The Scheduler
Function Library
Fast Dispatch latency
User-defined Data Objects and classes
Memory Management
Symmetric Multiprocessi ng
Components of Real Time Operating System

Fig. 1. The components of real time operating system

2.3 Propertise of Real Time Operating System

The time it takes the OS to receive and execute an application request is the unpredictability of an actual operating system jitter [80]. Hard RTOS systems have little jitter, while soft RTOS systems are quite jittery. There are a few important OS features to consider when it comes to data processing speed and jitter [81]:

- Reliability and stability: the operating system is unlikely to break or crash.
- Scalability: this is the operating system's ability, if more features are introduced, to increase its performance [82].
- Availability: This is an opportunity for the OS to actively process its request and not crash when requested.
- Usability: this is the phase of the development of an operating system.
- Security: This phase is not subject to an external attack by the operating system.

2.4 Scheduling Algorithms

Scheduling refers to the process of allocating available CPU resources to processes (s) [83]. This is a critical notion in the conception of multitasking, multiprocessing, and the real time operating systems [84]. Scheduler and dispatcher are used to do this. It is a decision-making process that involves allocating similar resources across several activities and time periods in order to accomplish many goals [85]. In a homogeneous/heterogeneous organization, resources and tasks may take on a variety of shapes. The jobs have been prioritized; each assignment is assigned a due date and an early completion date [86]. In real-time systems, the most crucial parameter is the deadline, which is defined as the time where the results should be supplied. As seen below, there are three unique types of timeline [85, 87].

- Soft Deadline: is that if findings are still valuable after the deadline has past, this is referred to as a soft deadline. This subcategory includes reservation systems.
- Firm deadline: This timeline is one that has no usefulness if it is missed. It is OK to miss a few deadlines on sometimes. These deadlines are often employed in systems that execute critical activities [88].
- Hard deadline: If a disaster occurs as a consequence of failing to meet the deadline, this is referred to be a hard deadline. This category includes systems that conduct mission-critical tasks such as air traffic control [89].

2.5 Scheduling Criteria

For CPU scheduling algorithms, several criteria have been presented; the characteristics used for comparison can make a considerable difference in selecting which technique is considered the most effective. Some of the requirements are below [90, 91]:

- CPU Usage: most of the time, the CPU would use the CPU the most instead of wasting every cycle of the CPU (Ideally I00 percent of the time). Because of a true system, CPU utilization should range between 40% (without charge) and 90% (heavily loaded).
- Performance: this indicates the whole number of processes per unit or the complete amount of labor per unit. This may range from l0/second to 1/hour depending on the particular processes [92].
- Time to turn: the needed amount of time for a certain procedure, i.e. the gap between the time of presentation and completion of the process [93].
- Waiting time: total of waiting time for a ready queue process waiting to be checked by the CPU.
- Load Average: the average number of processes waiting for CPU access [94].

3. RELATED Work

The following paragraphs summarize the work of different researchers in the topic of real-time processor scheduler from 2018 to 2020.

Teraiya and Shah [95], recommended that the LST and SJF be implemented in a real-time operating system that is not strictly real-time. Some algorithms were tested on a periodic work set, and the following results were achieved. They measured Success Ratio and Effective CPU Utilization under the identical settings and compared both algorithms. It is worth noting that the LST method works well under load but fails miserably under load. SJF is incapable of scheduling particular tasks even when the system is under stress, yet it functions well when the system is overloaded. On a large dataset, practical tests were undertaken. Each task set has between one and nine operations. The data set comprises 7500 task sets. Each process set uses between 0.5 and 5 percent of the CPU. It was validated against a 500-time unit to ensure that both algorithms were accurate.

Capodieci et al. [96], showcased the development of a sample real-time scheduler for GPU functions on an embedded System on a Chip (SoC) based on NVIDIA's cutting-edge Architectures used in self-driving cars. The scheduling acts as a software segmentation layer on top of the NVIDIA hypervisor, taking use of current-generation architectural features like as pixel-level and thread-level preemption. We were able to construct and test a GPU activity scheduler that uses the Earliest Deadline First (EDF) method. That provided Using a Constant Bandwidth Server, you may isolate your bandwidth. Using such an architecture (CBS). Our research focused on alternative programming paradigms for APIs computing, which enabled to describe CPU and GPU submittal of a request with more detailed scheduling data. A full experimental characterization is offered to demonstrate the considerable increase in scheduling of recurrent real time GPU operations.

Yang et al. [97], a hierarchy scheduling approach for DAG jobs with bounded deadlines was presented. Hierarchical scheduling separates the scheduling of computing resources from the scheduling of workloads, resulting in a significantly more effective resource sharing solution without regard for task kinds. We demonstrate the feasibility of our suggested hierarchical scheduling on a practical platform with an acceptable runtime overhead. Additionally, we conduct extensive tests to determine the schedule ability of our hierarchical scheduling system, and the findings indicate that our suggested technique performs admirably.

Cao and Bian [98], using a homogeneous multicore CPU, we suggested a DAG job scheduling technique. This is a refinement of the stretching method. The original method is enhanced by adding the ability to pick subtasks for conversion and by improving the computation of release time and deadline. This method has a substantially greater success rate than the original approach in terms of scheduling. Nevertheless, owing to the segmentation of tasks, a high number of task migrations and context switches occur throughout task scheduling, adding significant overhead and reducing task execution performance. The algorithm's next areas of optimization will aim to minimize task transfer and context switching.

Nasri et al. [99],The purpose of this study was to demonstrate how the Non-preemptive rescheduling may be made more useful by using a first-in-first-out (FIFO) scheduling strategy combined with a unique reduction tuning process. This methodology allows the FIFO to recreate a given viable scheduler, such as the one used by CW-EDF, resulting in a high degree of schedule ability and relatively modest runtime overheads. Memory overheads are also kept to a minimum by using a modest number of offsets per process. The runtime overhead, memory usage, and schedule ability ratio of the approach are all examined using a preliminary result on an Arduino Uno board.

Abeni et al. [100], proposed a new hierarchical scheduler for Linux that is optimized for container-based virtualization and can be utilized with LXC container that have multiple virtual CPUs. The provided scheduler is constructed by changing the real-time control group's mechanism in such a way that the SCHED DEADLINE policy is utilized to schedule each group's real-time run queues. Experiments demonstrate that a real-time application scheduled inside an LXC container using the new scheduler operates in the manner expected by previous theoretical CSF analysis. Our control groups scheduler proved to be simpler to setup and produced superior outcomes, while also having the ability to use less real-time computing capacity inside the system, so reducing runs.

Baital and Chakrabarti [101], proposed an improved scheduling algorithm in which random tasks with varying periodicity and execution time are generated at different time intervals. Energy consumption is a critical design consideration in real-time systems, particularly battery-powered systems. We extended our scheduling work to heterogeneous multicore systems (HMS) architectures, in which real-time tasks are distributed to the appropriate cores while still meeting the task deadline. They proved that the heterogeneous multicore scheduler paradigm may be utilized to create commercially accessible heterogeneous multicore processors. They validated the model using generated task sets and discovered that our model performs exceptionally well in all cases and significantly reduces the system's energy consumption when compared to some popular and novel scheduling techniques.

Chen et al. [102], develop a lightweight real time scheduling method that is based on job duplication, RTSATD, with objective of lowering both the time required to complete and the financial expense associated with cloud-based big-data workflow processing. RTSATD's performance is assessed using simulated and real-world processes. Testing findings indicate that the suggested method beats two current algorithms in terms of project length (by up to 28.73 %) and resource consumption (up to 46.31 percent).

Dauphin et al. [103], On parallel, heterogeneous, Non-Uniform Memory Architecture (NUMA) systems, a hybrid technique for scheduling and memory management of periodical dataflow apps was proven. The task ordering and memory allocation for each Processing Element are distributed and computed concurrently in Odyn. Additionally, He provides a strategy for preventing deadlocks induced by efforts to create buffers in memory that are larger than the available space. This method reduces the requirement for backtracking, which is a feature of dynamic scheduling algorithms. It is based on static calculation of exclusion relations among buffers in an application. They demonstrate Odyn's efficacy on a test bench that simulates the interactions of concurrent applications generated randomly. Additionally, they demonstrate their technique for avoiding deadlocks through a variety of use cases.

Riasetiawan and Ashari [104], proposed a schedule for the functioning of multiple landslide sensors Separately, data processing is transferred to an IoT device using FIFO and Round Robin scheduling. Only the performance of FIFO and Round Robin algorithms in scheduling incoming processes in real time on an IoT OS is discussed by the authors. Taking into account the waiting time and response time. The analysis is expected to result in a short response time and waiting time, allowing for the selection of an appropriate algorithm to complement the IoT architecture for landslide detection. The FIFO and Round Robin algorithms are implemented in the Raspberry Pi 3 Model B IoT device via Raspbian and Arches Ubuntu are two different operating systems. A 64-bit 64-bit ARM Cortex-A53 64-bit processor operating at 1.2GHz powers the Raspberry Pi 3 Model B.

Chen et al. [105], The results reported here establish the existence of Preemptive, fixed-priority real time systems now have a new scheduling side-channel (RTS). Systems that are examples of this kind include automobile systems, aviation system, electricity generation plants and industrial control systems. Notably, capturing this timing information is difficult because to schedule runtime changes, the system's incorporation of several interrelated activities, and the normal limitations (e.g deadline) inherent in the development of RTS. Schedule-Leak methods reveal how to exploit this side-channel successfully. On genuine operating systems, a comprehensive implementation is shown (in Real-time-Linux and Free RTOS). Schedule Leak's facts about the time period may considerably help other, more sophisticated attackers in fulfilling their objectives.

Malik et al. [106], suggested a novel For the proper execution and administration of real-time hard and soft activities in embedding IoT devices, an adaptable and intelligent scheduling mechanism is required. The reposed schedule method prioritizes the performance of critical real-time tasks with a high degree of priority activities above the distribution of CPU resources to potentially hungry, in overloaded instances, soft real-time processes. This was accomplished via the use of two astute deletions: Urgency Measure (UM) and Failure Measure (FM). By leveraging the available CPU unit for optimal CPU usage and rapid reaction times, the suggested methodology decreases the rate of tasks missed and jobs starved. The findings demonstrate that the suggested approach outperforms the other techniques in terms of task starvation rate reduction and CPU usage increase.

Doan and Tanaka [107], suggested a novel way of scheduling that is adaptable and intelligent for the effective execution and administration of real-time hard and soft tasks in embedded IoT devices. The reposed scheduling method prioritizes the execution of high-priority hard-real-time activities preceding the distribution of CPU resources to potentially hungry, in overloaded instances, soft real-time processes. This was accomplished via the use of two intelligent erasures: Urgency-Measure (UM) and Failure-Measure (FM). By leveraging the available CPU unit for optimal CPU usage and rapid reaction time, the suggested methodology decreases the rate of tasks missed and jobs starved. The findings demonstrate that the suggested approach outperforms the other techniques in terms of task starvation rate reduction and CPU usage increase.

D'souza and Rajkumar [108], presented the Cycle Tandem static frequency-scaling methodology for co-optimization of the CPU and hardware accelerators operating frequencies. Consider different energy management situations where the accelerator or CPU frequency may or may not be configurable, and present the Cycle Solo family of algorithms for such situations based on practical considerations of real-world platforms. Additionally, when multi-core processors are utilized in combination with hardware accelerators, partitioning solutions should be investigated to lower operation frequency. Experiments suggest that some strategies may result in large energy savings. Additionally, we offer a case study using the NVIDIA TX2 embedded platform to demonstrate the energy reductions that our suggested solutions may achieve.

Nguyen et al. [109], suggested Real time scheduling of Household Appliances' Quality learning, a well-known value iterative reinforcement learning methodology, is used to calculate operational time (RSOTHA-QL). The RSOTHA-QL procedure is a two-step procedure. The first step involves Q learning agents engaging with the smart home system with the purpose of receiving a reward. Additionally, the incentive value is used to arrange the operating duration of home App in next step, ensure that energy consumption is kept to a minimum. The second phase addresses discontent caused by the scheduling of the operating time of the home user's domestic appliances by dividing them in to Three categories: 1) deferrable, 2) non-deferrable, and3) controlled. It is discovered that the operating period of household appliances is effectively planned in order to considerably minimize energy usage and user discontent.

Khan et al. [110], demonstrated the efficacy of our suggested DRL algorithm, for scheduling energy harvesting times during UAV-assisted ,D2D communication. The terms of EE and complexity, offered approaches outperform benchmarks. The energy harvesting time scheduling game may be resolving extremely fast by exploiting the benefits of deep learning. The findings indicate that the DRL method may be a viable solution for a real time app, given the energy storage constraints and the times constraints associated with flying time-constrained UAVs.

Chen et al. [111], has removed a significant constraint associated with newly established dependency graph approaches, namely just one crucial section per task (DGA). Under terms of computational complexity, shown that even in extremely limited settings, the multiprocessor synchronization issue is NP-complete. Suggest a Provide the approximation ratio(s) for the resulting computation time based on a systematic proposed design based on the DGA by leveraging current algorithms designed for job shop scheduling. The results of the evaluation in Section 6.2 show that the methodology is quite effective for real-time task systems based on frames.

Utkarsh et al. [112], a completely distributed model-predictive and computational intelligence program that enables micro grid devices to function autonomously with little communication exchange, hence eliminating the need for a central controller. The proposed distributed algorithm's convergence features are evaluated and compared to those of a state-of-the-art distributed algorithm, and numerical simulations for various situations are conducted to demonstrate that the proposed distributed approach may be implemented to real-world micro grids.

Lee et al. [113], It is advised that a real time schedule study of transiently powered CPUs with a NVMs be performed. To begin, quantitatively analyze the energy harvester's charging and discharging behaviors and extract the system's compute capabilities in the time interval domain. Then, using real time calculus, we establish whether or not a particular multi-task workload can be scheduled using the earliest possible deadline first (EDF) or fixed-priority (FP) scheduling rules. Furthermore, the work examined how the threshold voltage parameter selection influences schedulability and then developed a practical threshold selection technique to improve schedulable. Comprehensive simulations are used to validate the suggested technique's efficacy. In comparison to the naïve selection approach, the strategy consistently improves schedule ability across a range of workloads.

El Ghor et al. [114], The challenge of real-time scheduling of many processors at the systems with an It was investigated if an energy reservoir

Table 1. Summary of literature review related to Scheduling Algorithms for Real Time Operating Systems

No.	Ref. and Year	RTOS Type	Operating System	Significant Results	Future Work Suggestions
1	[95] 2018	Soft RTOS	Real time operating system	Data Set Includes 7500 task sets. Each procedure is an individual job. processor load	In the coming, we may suggest a new method that incorporates LST and SJF. Load and demand will make it function well.
2	[96] 2018	Soft RTOS	embedded System	For quickly executing aperiodic jobs, we give them a null, or small, deadline. This allows the aperiodic jobs to preempt already executing jobs.	In the near, we hope to introduce more complex reclamation techniques for GPUs' excess bandwidth, which will make more flexible selections of GPU task budgets possible.
3	[97] 2018	Hard RTOS	A Hierarchical Scheduling Approach	Our hierarchical scheduling implementation was assessed and findings showed it to be promising.	Future directions for this study abound. Let's first look into scheduling many DAG jobs on several virtual platforms under various platforms. Also, DAG task scheduling models with resource sharing is a worthwhile consideration. Secondly, we want to use power scheduling techniques for DAG.
4	[98] 2020	Hard RTOS	Multi-core Processors	This method's scheduling success rate is much higher than the original approach.	Next up are tasks and context switching for the algorithm.
5	[99] 2018	Hard RTOS	Arduino board	With respect to code size, FIFO-OT scheduler footprints are likewise less than OE's.	The aggregate minimum amount of offsets for all jobs and extended systems with release jitter.
6	[100] 2019	Soft RTOS	Linux Kernel	Experimental findings reveal that real-time LXC-based scheduling performed under the new scheduler behaved as expected by current CSF analyses.	In the upcoming, we want to experiment with the suggested scheduler in a multi-CPU container-based simultaneous real-time activity setting.
7	[101] 2019	Hard RTOS	Multicore systems	Our approach works very well in all instances, and greatly decreases energy usage when compared to certain popular and novel schedules.	The additional feature will help implement dependent task sets.
8	[102] 2020	Hard RTOS	Geo-Distributed Clouds	The testing findings show that the suggested algorithm offers 28.73% more completion time reduction and 46.31% greater resource usage than the other alternatives.	Therefore, fault-tolerant huge data processing scheduling should be one of the noteworthy approaches to pursue.
9	[103] 2019	RTOS	Dataflow Applications	Authors show the usefulness of Odyn on a test bench that replicates random program interactions We illustrate the stalemate avoidance strategy in several scenarios.	A suggests research project is to calculate the appropriate memory assignments for PEs, depending on their performance requirements.
12	[106] 2019	Hard and Soft Real Time Hybrid	IoT Devices Embedded	The study found that the suggested approach outperformed the other techniques, considerably lowering task starvation and boosting CPU usage.	One of the drawbacks of the suggested solution is that it requires time to collect sufficient training data.
13	[107] 2018	Soft RTOS	Real-time embedded systems.	The outcomes from our simulation demonstrate that our technique has decreased time complexity while still maintaining schedulability, task preemption, and task migration.	In addition, using a hardware accelerator may minimize the runtime overhead of the method. This inquiry is in the realm of practice and segmented near-optimal techniques.

No.	Ref. and Year	RTOS Type	Operating System	Significant Results	Future Work Suggestions
14	[108] 2018	Hard RTOS	Cyber-physical systems	Lastly, we implement the Cycle Solo and Cycle Tandem algorithms to full-partitioned multi-core CPUs, where all CPU cores should be tuned to the same rate.	Multi-core systems are where the rate of each processor core may be separately adjusted in the next. Systems which feature numerous accelerators may also be considered.
15	[109] 2020	Soft RTOS	neural networks	The findings show that the DRL method may be applicable for real-time applications even if energy storage and flight duration are limited.	In the approach, we will handle more complex challenges by simultaneously optimizing power allocation, trajectory planning, and numerous UAV scenarios.
16	[110] 2018	Hard RTOS	Smart Home Appliances	The suggested scheduling technique improves on the LST based scheduling when it comes to decreasing energy use and lowering unhappiness for the house us.	Future research will use deep Q knowledge and artificial neural networks to simulate a complex multiple-home user situation.
17	[111] 2020	Soft RTOS	Multicore systems	Using created task sets, the model was verified and shown to perform brilliantly and dramatically lower energy usage in comparison to a few popular and novel scheduling approaches.	Our next projects will be focused on validating the scheduling model, ensuring it is fault tolerant and reliability conscious.
18	[112] 2018	Soft RTOS	Network of Smart Micro grids	in the simulation game, there are huge benefits to reactive power trading, including dynamic price adjustments between micro grids.	Not mentioned
19	[113] 2020	Hard RTOS	Wireless Sensor Network	This article examines the schedulability of transiently supplied CPUs with NVMs in real time. Quantified analysing the behaviour of the energy harvester across time intervals permits real-time multi-task execution time analysis using the RTC.	Additionally, minimizing the threshold voltage searching overhead without approximation remains a future studies objective.
20	[114] 2018	Hard RTOS	Ambient energy	Our method delivers substantial performance improvement when compared to EDF.	The next study will concentrate on an evaluation of the performance of EH-RA comparison to EDF, which uses the bin-packing methodology.

could be recharged by an external source.. We are particularly interested in energy-efficient partitioning for periodic real-time applications on a homogenous multi-core architecture, taking into account both timing and energy needs. Assume that the optimum scheduler is employed on each core of the design, namely the Earliest Deadline - Harvesting (ED-H). The purpose is to provide realistic to achieve the intended absence of both energy famine and deadline violation, a segmentation method based on the real-time execution of tasks given to sensor nodes utilizing actual energy recovery data was developed. To accomplish this, they presented an Energy Harvesting Reasonable allocation (EH-RA) method that is equivalent to the conventional bin-packing methodology in terms of both temporal limitations and energy awareness. Experimental data indicate that our technique can significantly outperform EDF in terms of performance.

4. DISCUSSION

Additionally, it is necessary to emphasize that in some real-world applications, end-to-end time limitations apply to calculations that span many processing locations. We will present some of the most prominent programming algorithms used in programming for the CPU. They are not all suited for usage in embedded systems in real-time. Currently, non-preventative planning, round timing and preemptive priority planning are the most often used Algorithms in real RTOS. The findings reveal that an overview of the linked research is provided in the table below. The table gives the reference name and the corresponding year of the investigations (2018, 2019 and 2020), the kind of RTOS system, the system used by them and the results obtained from research and recommendations or future study where these are conducted.

5. CONCLUSION

Real-time operating system enables real-time apps achieve their deadline through scheduling, the scheduling approach is it the beating hearts of every Real-Time System, which controls every job execution to prevent overlap. Real-time operating systems are fundamental to the functioning of numerous technologies and technologies that are crucial to our everyday lives. Additionally, these systems have the capacity to execute the software necessary and to give temporal precision, which means that they are able to assure that the termination of a program or task execution is related to a time interval or exact instant in time. This characteristic is very important in many applications where any delay in implementing the program or any delay in the reaction time results in adverse effects for the system, such as automated control systems in industrial equipment and different transportation modes. In this review paper about real time operating system a total of 20 papers we used in a related works from years 2018 to 2020 based of the findings the number of the papers about RTOS increased from 2020. Most of the articles about the hard RTOS type, and many of the related work used in embedded systems.

DISCLAIMER

The products used for this research are commonly and predominantly use products in our area of research and country. There is absolutely no conflict of interest between the authors and producers of the products because we do not intend to use these products as an avenue for any litigation but for the advancement of knowledge. Also, the research was not funded by the producing company rather it was funded by personal efforts of the authors.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Sadeeq MA, Zeebaree S. Energy management for internet of things via distributed systems. Journal of Applied Science and Technology Trends. 2021;2:59-71.
2. Hambarde P, Varma R, Jha S. The survey of real time operating system: RTOS. 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies. 2014;34-39.
3. Abdullah RM, Ameen SY, Ahmed DM, Kak SF, Yasin HM, Ibrahim IM, et al. Paralinguistic Speech Processing: An Overview. Asian Journal of Research in Computer Science. 2021;34-46.
4. Ibrahim IM, Ameen SY, Yasin HM, Omar N, Kak SF, Rashid ZN, et al. Web server performance improvement using dynamic load balancing techniques: A review. Asian Journal of Research in Computer Science. 2021;47-62.

5. Omer MA, Zeebaree SR, Sadeeq MA, Salim BW, Mohsin Sx, Rashid ZN, et al. Efficiency of malware detection in android system: A survey. Asian Journal of Research in Computer Science. 2021;59-69.
6. Donga J, Holia M. An analysis of scheduling algorithms in real-time operating system. International Conference on Inventive Computation Technologies. 2019;374-381.
7. Maulud DH, Zeebaree SR, Jacksi K, Sadeeq MAM, Sharif KH. State of art for semantic analysis of natural language processing. Qubahan Academic Journal. 2021;1:21-28.
8. Ahmed DM, Ameen SY, Omar N, Kak SF, Rashid ZN, Yasin HM, et al. A state of art for survey of combined iris and fingerprint recognition systems. Asian Journal of Research in Computer Science. 2021;18-33.
9. Brucker P, Heitmann S, Hurink J, Nieberg T. Job-shop scheduling with limited capacity buffers. OR spectrum. 2006;28:151-176.
10. Ibrahim BR, Khalifa FM, Zeebaree SR, Othman NA, Alkhayyat A, Zebari RR, et al. Embedded system for eye blink detection using machine learning technique. 2021 1st Babylon International Conference on Information Technology and Science (BICITS). 2021;58-62.
11. Maulud DH, Ameen SY, Omar N, Kak SF, Rashid ZN, Yasin HM, et al. Review on natural language processing based on different techniques. Asian Journal of Research in Computer Science. 2021;1-17.
12. Salih AA, Ameen SY, Zeebaree SR, Sadeeq MA, Kak SF, Omar N, et al. Deep learning approaches for intrusion detection. Asian Journal of Research in Computer Science. 2021;50-64.
13. Hassan RJ, Zeebaree SR, Ameen SY, Kak SF, Sadeeq MA, Ageed ZS, et al. State of art survey for iot effects on smart city technology: Challenges, opportunities, and solutions. Asian Journal of Research in Computer Science. 2021;32-48.
14. Zebari S, Yaseen NO. Effects of parallel processing implementation on balanced load-division depending on distributed memory systems. J. Univ. Anbar Pure Sci. 2011;5:50-56.
15. Hasan DA, Zeebaree SR, Sadeeq MA, Shukur HM, Zebari RR, Alkhayyat AH. Machine learning-based diabetic retinopathy early detection and classification systems-A survey. 2021 1st Babylon International Conference on Information Technology and Science (BICITS). 2021;16-21.
16. Sadeeq MM, Abdulkareem NM, Zeebaree SR, Ahmed DM, Sami AS, Zebari RR. IoT and Cloud computing issues, challenges and opportunities: A review. Qubahan Academic Journal. 2021;1:1-7.
17. Yahia HS, Zeebaree SR, Sadeeq MA, Salim NO, Kak SF, Adel AZ, et al. Comprehensive survey for cloud computing based nature-inspired algorithms optimization scheduling. Asian Journal of Research in Computer Science. 2021;1-16.
18. Turci LdO. Real-time operating system freertos application for fire alarm project in reduced scale. International Journal of Computing and Digital Systems. 2017;6:197-204.
19. Hasan DA, Hussan BK, Zeebaree SR, Ahmed DM, Kareem OS, Sadeeq MA. The impact of test case generation methods on the software performance: A review. International Journal of Science and Business. 2021;5:33-44.
20. Ageed ZS, Zeebaree SR, Sadeeq MM, Kak SF, Rashid ZN, Salih AA, et al. A survey of data mining implementation in smart city applications. Qubahan Academic Journal. 2021;1:91-99.
21. Jacksi K, Ibrahim RK, Zeebaree SR, Zebari RR, Sadeeq MA. Clustering documents based on semantic similarity using HAC and K-mean algorithms. 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;205-210.
22. Ageed ZS, Zeebaree SR, Sadeeq MA, Abdulrazzaq MB, Salim BW, Salih AA, et al. A state of art survey for intelligent energy monitoring systems," Asian Journal of Research in Computer Science. 2021;46-61.
23. Jijo BT, Zeebaree SR, Zebari RR, Sadeeq MA, Sallow AB, Mohsin S, et al. A comprehensive survey of 5G mm-wave technology design challenges. Asian Journal of Research in Computer Science. 2021;1-20.
24. Arsinte R. Real time operating system options in connected embedded equipment for distributed data acquisition. Carpathian

Journal of Electronic and Computer Engineering. 2018;11:35-58.

25. Sadeeq MA, Abdulazeez AM. Neural networks architectures design, and applications: A review. 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;199-204.
26. Ageed ZS, Ibrahim RK, Sadeeq M. Unified ontology implementation of cloud computing for distributed systems. Current Journal of Applied Science and Technology. 2020;82-97.
27. Zeebaree S, Ameen S, Sadeeq M. Social media networks security threats, risks and recommendation: A case study in the kurdistan region. International Journal of Innovation, Creativity and Change. 2020;13:349-365.
28. Sulaiman MA, Sadeeq M, Abdulraheem AS, Abdulla AI. Analyzation study for gamification examination fields. Technol. Rep. Kansai Univ. 2020;62:2319-2328.
29. Sadeeq M, Abdulla AI, Abdulraheem AS, Ageed ZS. Impact of electronic commerce on enterprise business. Technol. Rep. Kansai Univ. 2020;62:2365-2378.
30. Salih A, Zeebaree ST, Ameen S, Alkhyyat A, Shukur HM. A survey on the role of artificial intelligence, machine learning and deep learning for cybersecurity attack detection. 2021 7th International Engineering Conference “Research & Innovation amid Global Pandemic"(IEC). 2021;61-66.
31. Alzakholi O, Shukur H, Zebari R, Abas S, Sadeeq M. Comparison among cloud technologies and cloud performance. Journal of Applied Science and Technology Trends. 2020;1:40-47.
32. Abdullah DM, Ameen SY, Omar N, Salih AA, Ahmed DM, Kak SF, et al. Secure data transfer over internet using image steganography. Asian Journal of Research in Computer Science. 2021;33-52.
33. Ageed Z, Mahmood MR, Sadeeq M, Abdulrazzaq MB, Dino H. Cloud computing resources impacts on heavy-load parallel processing approaches. IOSR Journal of Computer Engineering (IOSR-JCE). 2020;22:30-41.
34. Kareem FQ, Ameen SY, Salih AA, Ahmed DM, Kak SF, Yasin HM, et al. SQL injection attacks prevention system technology. Asian Journal of Research in Computer Science. 2021;13-32.
35. Sallow A, Zeebaree S, Zebari R, Mahmood M, Abdulrazzaq M, Sadeeq M. Vaccine tracker. SMS reminder system: Design and implementation; 2020.
36. Malallah H, Zeebaree SR, Zebari RR, Sadeeq MA, Ageed ZS, Ibrahim IM, et al. A comprehensive study of kernel (issues and concepts) in different operating systems. Asian Journal of Research in Computer Science. 2021;16-31.
37. Sadeeq MA, Zeebaree SR, Qashi R, Ahmed SH, Jacksi K. Internet of things security: A survey. in 2018 International Conference on Advanced Science and Engineering (ICOASE). 2018;162-166.
38. Ismael HR, Ameen SY, Kak SF, Yasin HM, Ibrahim IM, Ahmed AM, et al. Reliable communications for vehicular networks. Asian Journal of Research in Computer Science. 2021;33-49.
39. Yasin HM, Zeebaree SR, Sadeeq MA, Ameen SY, Ibrahim IM, Zebari RR, et al. IoT and ICT based smart water management, monitoring and controlling system: A review. Asian Journal of Research in Computer Science. 2021;42-56.
40. Abdulla AI, Abdulraheem AS, Salih AA, Sadeeq M, Ahmed AJ, Ferzor BM, et al. Internet of things and smart home security. Technol. Rep. Kansai Univ. 2020;62:2465-2476.
41. Abdulraheem AS, Salih AA, Abdulla AI, Sadeeq M, Salim N, Abdullah H, et al. Home automation system based on IoT; 2020.
42. Salih AA, Zeebaree S, Abdulraheem AS, Zebari RR, Sadeeq M, Ahmed OM. Evolution of mobile wireless communication to 5G revolution. Technology Reports of Kansai University. 2020;62:2139-2151.
43. Dino HI, Zeebaree S, Salih AA, Zebari RR, Ageed ZS, Shukur HM, et al. Impact of process execution and physical memory-spaces on OS performance. Technology Reports of Kansai University. 2020;62:2391-2401.
44. Hamdi SJ, Ibrahim IM, Omar N, Ahmed OM, Rashid ZN, Ahmed AM, et al. A comprehensive study of malware detection in android operating systems.
45. Ibrahim IM. Task scheduling algorithms in cloud computing: A review. Turkish Journal of Computer and Mathematics Education (TURCOMAT). 2021;12:1041-1053.
46. Ageed ZS, Ahmed AM, Omar N, Kak SF, Ibrahim IM, Yasin HM, et al. A state of art survey of nano technology:

Implementation, Challenges, and future trends.
47. Abdulazeez AM, Zeebaree SR, Sadeeq MA. Design and implementation of electronic student affairs system. Academic Journal of Nawroz University. 2018;7:66-73.
48. Abdulqadir MM, Salih AA, Ahmed OM, Hasan DA, Haji LM, Ahmed SH, et al. A comprehensive study of caching effects on fog computing performance.
49. Yazdeen AA, Zeebaree SR, Sadeeq MM, Kak SF, Ahmed OM, Zebari RR. FPGA implementations for data encryption and decryption via concurrent and parallel computation: A review. Qubahan Academic Journal. 2021;1:8-16.
50. Ageed ZS, Zeebaree SR, Sadeeq MM, Kak SF, Yahia HS, Mahmood MR, et al. Comprehensive survey of big data mining approaches in cloud systems. Qubahan Academic Journal. 2021;1:29-38.
51. Zeebaree S, Zebari RR, Jacksi K. Performance analysis of IIS10. 0 and Apache2 Cluster-based Web Servers under SYN DDoS Attack. TEST Engineering & Management. 2020;83:5854-5863.
52. Abdulrahman LM, Zeebaree SR, Kak SF, Sadeeq MA, Adel AZ, BW. Salim, et al. A state of art for smart gateways issues and modification. Asian Journal of Research in Computer Science. 2021;1-13.
53. Sallow AB, Sadeeq M, Zebari RR, Abdulrazzaq MB, Mahmood MR, Shukur HM, et al. An investigation for mobile malware behavioral and detection techniques based on android platform. IOSR Journal of Computer Engineering (IOSR-JCE). 2020;22:14-20.
54. Abdulqadir HR, Zeebaree SR, Shukur HM, Sadeeq MM, Salim BW, Salih AA, et al. A study of moving from cloud computing to fog computing. Qubahan Academic Journal. 2021;1:60-70.
55. Dino H, Abdulrazzaq MB, Zeebaree S, Sallow AB, Zebari RR, Shukur HM, et al. Facial expression recognition based on hybrid feature extraction techniques with different classifiers. TEST Engineering & Management. 2020;83:22319-22329.
56. AL-Zebari A, Zeebaree S, Jacksi K, Selamat A. ELMS–DPU ontology visualization with Protégé VOWL and Web VOWL. Journal of Advanced Research in Dynamic and Control Systems. 2019;11:478-85.
57. Zeebaree A, Adel A, Jacksi K, Selamat A. Designing an ontology of E-learning system for duhok polytechnic university using protégé OWL tool. J Adv Res Dyn Control Syst. 2019;11:24-37.
58. Jader OH, Zeebaree S, Zebari RR. A state of art survey for web server performance measurement and load balancing mechanisms. International Journal of Scientific & Technology Research. 2019;8:535-543.
59. Adel AZ, Zebari S, Jacksi K. Football ontology construction using oriented programming. Journal of Applied Science and Technology Trends. 2020;1:24-30.
60. Zeebaree S, Zebari RR, Jacksi K, Hasan DA. Security approaches for integrated enterprise systems performance: A Review. Int. J. Sci. Technol. Res. 2019;8.
61. Selamat SAAZA. Electronic learning management system based on semantic web technology: A review. Int. J. Adv. Electron. Comput. Sci. 2017;4:1-6.
62. Abdullah RM, Abdulazeez AM, Al-Zebari A. Machine learning algorithm of intrusion detection system. Asian Journal of Research in Computer Science. 2021;1-12.
63. Zebari IM, Zeebaree SR, Yasin HM. Real time video streaming from multi-source using client-server for video distribution," in 2019 4th Scientific International Conference Najaf (SICN). 2019;109-114.
64. Shukur H, Zeebaree SR, Ahmed AJ, Zebari RR, Ahmed O, Tahir BSA, et al. A state of art survey for concurrent computation and clustering of parallel computing for distributed systems. Journal of Applied Science and Technology Trends. 2020;1:148-154.
65. Ibrahim BR, Zeebaree SR, Hussan BK. Performance measurement for distributed systems using 2TA and 3TA based on OPNET principles. Science Journal of University of Zakho. 2019;7:65-69.
66. Tahir B, Ali Saktioto J, Fadhali M, Rahman R, Ahmed A. A study of FBG sensor and electrical strain gauge for strain measurements. Journal of optoelectronics and advanced materials. 2008;10:2564-2568.
67. Zeebaree SR, Sallow AB, Hussan BK, Ali SM. Design and simulation of high-speed parallel/sequential simplified DES code breaking based on FPGA. 2019 International Conference on Advanced

Science and Engineering (ICOASE). 2019;76-81.
68. Harki N, Ahmed A, Haji L. CPU scheduling techniques: A review on novel approaches strategy and performance assessment. Journal of Applied Science and Technology Trends. 2020;1:48-55.
69. Ahmed A, Ahmed O. Correlation pattern among morphological and biochemical traits in relation to tillering capacity in sugarcane (*Saccharum* Spp). Acad J Plant Sci. 2012;5:119-122.
70. Zebari DA, Haron H, Zeebaree SR, Zeebaree DQ. Multi-Level of DNA Encryption Technique Based on DNA Arithmetic and Biological Operations. 2018 International Conference on Advanced Science and Engineering (ICOASE). 2018;312-317.
71. Ahmed AJ, Mohammed FH, Majedkan NA. An evaluation study of an E-learning course at the Duhok Polytechnic University: A case study. Journal of Cases on Information Technology (JCIT). 2022;24:1-11.
72. Ahmed O, Geraldes R, Ahmed A, DeLuca G, Palace J. Multiple sclerosis and the risk of venous thrombosis: a systematic review. MUltiple Sclerosis Journal. 2017;757-758.
73. Juven V. Lightweight Event-driven real-time operating system for resource constrained connectivity; 2017.
74. Gracioli G, Fröhlich AA, Pellizzoni R, Fischmeister S. Implementation and evaluation of global and partitioned scheduling in a real-time OS. Real-Time Systems. 2013;49:669-714.
75. Holman P, Anderson JH. Adapting Pfair scheduling for symmetric multiprocessors. Journal of Embedded Computing. 2005;1:543-564.
76. Chapin SJ, Spafford EH. Support for implementing scheduling algorithms using MESSIAHS. Scientific Programming. 1994;3:325-340.
77. Kato S, Ishikawa Y, Rajkumar RR. CPU scheduling and memory management for interactive real-time applications. Real-Time Systems. 2011;47:454-488.
78. Yousefi H, Malekimajd M, Ashouri M, Movaghar A. Fast aggregation scheduling in wireless sensor networks. IEEE Transactions on Wireless Communications. 2015;14:3402-3414.
79. Bjørk J, de Boer FS, Johnsen EB, Schlatte R, Tarifa SLT. User-defined schedulers for real-time concurrent objects. Innovations in Systems and Software Engineering. 2013;9:29-43.
80. Salim NO, Abdulazeez AM. Human diseases detection based on machine learning algorithms: A review. International Journal of Science and Business. 2021;5:102-113.
81. Khan S. Real-Time operating system (RTOS) with Different application: A systematic mapping. European Journal of Engineering and Technology Research. 2021;6:100-103.
82. Yasin HM, Zeebaree SR, Zebari IM. Arduino based automatic irrigation system: Monitoring and SMS controlling. 2019 4th Scientific International Conference Najaf (SICN). 2019;109-114.
83. Zeebaree S, Yasin HM. Arduino based remote controlling for home: Power saving, security and protection. International Journal of Scientific & Engineering Research. 2014;5:266-272.
84. Salim NO, Zeebaree SR, Sadeeq MA, Radie A, Shukur HM, Rashid ZN. Study for food recognition system using deep learning. Journal of Physics: Conference Series. 2021;012014.
85. Khera I, Kakkar A. Comparative study of scheduling algorithms for real time environment. International Journal of Computer Applications. 2012;44:5-8.
86. Salim NO, Abdulazeez AM. Science and business. International Journal. 5:102-113.
87. Du D. Scheduling Algorithms: JSTOR; 2008.
88. Eesa AS, Sadiq S, Hassan M, Orman Z. Rule generation based on modified cuttlefish algorithm for intrusion detection system. Uludağ University Journal of The Faculty of Engineering. 2021;26:253-268.
89. Eesa AS. Optimization algorithms for intrusion detection system: A review. International Journal of Research-GRANTHAALAYAH. 2020;8:217-225.
90. Mahmud N, Afrin S, Rahman F, Monirujjaman M. An application based improved round robin CPU scheduling for real time operating system. Brac University; 2017.
91. Haji SH, Zeebaree SR, Saeed RH, Ameen SY, Shukur HM, Omar N, et al. Comparison of software defined networking with traditional networking. Asian Journal of Research in Computer Science. 2021;1-18.
92. Zeebaree S, Zebari I. Multilevel client/server peer-to-peer video

broadcasting system. International Journal of Scientific & Engineering Research. 2014;5:260-265.

93. Kareem FQ, Zeebaree SR, Dino HI, Sadeeq MA, Rashid ZN, Hasan DA, et al. A survey of optical fiber communications: Challenges and processing time influences. Asian Journal of Research in Computer Science. 2021;48-58.

94. Abdullah SMSA, Ameen SYA, Sadeeq MA, Zeebaree S. Multimodal emotion recognition using deep learning. Journal of Applied Science and Technology Trends. 2021;2:52-58.

95. Teraiya J, Shah A. Comparative study of LST and SJF scheduling algorithm in soft real-time system with its implementation and analysis. 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2018;706-711.

96. Capodieci N, Cavicchioli R, Bertogna M, Paramakuru A. Deadline-based scheduling for gpu with preemption support. 2018 IEEE Real-Time Systems Symposium (RTSS). 2018;119-130.

97. Yang T, Deng Q, Sun L. Building real-time parallel task systems on multi-cores: A hierarchical scheduling approach. Journal of Systems Architecture. 2019;92:1-11.

98. Cao S, Bian J. Improved DAG tasks stretching algorithm based on multi-core processors. 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS). 2020;18-21.

99. Nasri M, Davis RI, Brandenburg BB. FIFO with offsets: High schedulability with low overheads. in 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). 2018;271-282.

100. Abeni L, Balsini A, Cucinotta T. Container-based real-time scheduling in the linux kernel. ACM SIGBED Review. 2019;16:33-38.

101. Baital K, Chakrabarti A. Various approaches for high throughput and energy efficient scheduling of real-time tasks in multicore systems. 2019 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS). 2019;402-405.

102. Chen H, Wen J, Pedrycz W, Wu G. Big data processing workflows oriented real-time scheduling algorithm using task-duplication in geo-distributed clouds. IEEE Transactions on Big Data. 2018;6:131-144.

103. Dauphin B, Pacalet R, Enrici A, Apvrille L. Odyn: Deadlock Prevention and Hybrid Scheduling Algorithm for Real-Time Dataflow Applications. 2019 22nd Euromicro Conference on Digital System Design (DSD). 2019;88-95.

104. Riasetiawan M, Ashari A. Performance analysis of FIFO and Round robin scheduling process algorithm in IoT Operating system for collecting landslide data. 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA). 2020;63-68.

105. Chen CY, Mohan S, Pellizzoni R, Bobba RB, Kiyavash N. A novel side-channel in real-time schedulers. 2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). 2019;90-102.

106. Malik S, Ahmad S, Ullah I, Park DH, Kim D. An adaptive emergency first intelligent scheduling algorithm for efficient task management and scheduling in hybrid of hard real-time and soft real-time embedded IoT systems. Sustainability. 2019;11:2192.

107. Doan D, Tanaka K. A novel task-to-processor assignment approach for optimal multiprocessor real-time scheduling. 2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). 2018;101-108.

108. D'souza S, Rajkumar R. Cycletandem: Energy-saving scheduling for real-time systems with hardware accelerators. 2018 IEEE Real-Time Systems Symposium (RTSS). 2018;94-106.

109. Nguyen KK, Vien NA, Nguyen LD, Le MT, Hanzo L, Duong TQ. Real-time energy harvesting aided scheduling in UAV-assisted D2D networks relying on deep reinforcement learning. IEEE Access. 2020;9:3638-3648.

110. Khan M, Seo J, Kim D. Real-time scheduling of operational time for smart home appliances based on reinforcement learning. IEEE Access. 2020;8:116520-116534.

111. Chen JJ, Shi J, von der Brüggen G, Ueter N. Scheduling of real-time tasks with multiple critical sections in multiprocessor systems. arXiv preprint arXiv:2007.08302; 2020.

112. Utkarsh K, Srinivasan D, Trivedi A, Zhang W, Reindl T. Distributed model-predictive

real-time optimal operation of a network of smart microgrids," IEEE Transactions on Smart Grid. 2018;10:2833-2845.

113. Lee D, Jung H, Yang H. Real-time schedulability analysis and enhancement of transiently powered processors with nvms. IEEE Transactions on Computers. 2020;70:372-383.

114. El Ghor H, Chetto M, El Osta R. Multiprocessor Real-Time Scheduling for Wireless Sensors Powered by Renewable Energy Sources. 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA). 2018;1-6.

Peer-review history:
The peer review history for this paper can be accessed here:
https://www.sdiarticle4.com/review-history/73758