Simulation and Comparative Study of Various Task Scheduling Algorithms

Umang Jain
19MCA0164
SITE, VIT Vellore
Umangj107@gmail.com

Rahul Sharma
19MCA0174
SITE, VIT Vellore
Rahul.Sharma2019@vitstudent.ac.in

Akshit
19MCA0020
SITE, VIT Vellore
akshitpanwar1234@gmail.com

Prof. Uma Maheswari G
Associate Professor
SITE, VIT Vellore
g.umamaheswari@vit.ac.in

Abstract: **Operating System handles all the tasks of the user successfully and efficiently. The operating System makes use of certain Task Scheduling Algorithms in order to schedule the various lined up tasks according to some criteria. There had been a number of Task Scheduling Algorithms proposed by different researchers mastering in this field and all have proved efficient in some or the other way. The main purpose of this paper is to simulate the various Task Scheduling algorithms in order to better understand their working as well as conducting a comparative study of each algorithm against others.**

Keywords: Operating systems, Task scheduling, Task Scheduling Algorithms, First Come First Serve, Round Robin, Shortest Job First, Shortest Remaining Time First, Priority Scheduling.

I. INTRODUCTION

Operating System (OS) is system software which acts as an interface between a user and the computer hardware. OS is also known as resource manager because its prime responsibility is to manage the resources of the computer system. Scheduling is a fundamental and most important OS Function which is essential to an operating system's design. Scheduling refers to set of rules, policies and mechanism that govern the order in which resource is allocated to the various processes and the work is to be done. The scheduling is a methodology of managing multiple queues of processes in order to minimize delay and to optimize performance of the system. A scheduler is an OS module that implements the scheduling policies. Its primary objective is to optimize the system performance according to the criteria set by the system designers. It selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.

Types of scheduling:

First Come First Served (FCFS): - FCFS, also known as First in First out (FIFO), is the simplest scheduling policy. Arriving jobs are inserted into the tail (rear) of the ready queue and the process to be executed next is removed from the head (front) of the queue. FCFS performs better for long jobs. Relative Importance of jobs measured only by arrival time (poor choice) [1]. A long CPU bound job may hog the CPU and May force shorter (or I/O bound) jobs to wait prolonged periods. This in turn may lead to a Lengthy queue of ready jobs, and thence to the ''convoy effect.''

Shortest Job First (SJF): - SJF policy selects the job with the shortest (expected) processing time first. Shorter jobs are always executed before long jobs. One major difficulty with SJF is the need to know or estimate the processing time of each job Also, long running jobs may starve, because the CPU has a steady supply of short jobs. Round-Robin (RR) reduces the penalty that short jobs suffer with FCFS by pre-empting running jobs periodically. The CPU suspends the current job when the reserved quantum (time-slice) is exhausted. The job is then put at the end of the ready queue if not yet completed. The critical issue with the RR policy is the length of the quantum. If it is too short, then the CPU will be spending more time on context switching. Otherwise, interactive processes will suffer II. Scheduling Criteria Different CPU scheduling algorithms have different properties and the choice of a particular algorithm May favours one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms.

Round robin :- It is an arrangement of choosing all elements in a group equally in some rational order, usually from the top to the bottom of a list and then starting again at the top of the list and so on. A simple way to think of round robin is that it is about "taking turns." Used as an adjective, round robin becomes "round-robin."
In computer operation, one method of having different program process take turns using the resources of the computer is to limit each process to a certain short time period, then suspending that process to give another process a turn (or "time-slice"). This is often described as round-robin process scheduling.

Priority Scheduling :- Priority Scheduling Each process is assigned a priority. The ready list contains an entry for each process ordered by its priority. The process at the beginning of the list (highest priority) is picked first. A variation of this scheme allows pre-emption of the current process when a higher priority process Arrives. Another variation of the policy adds an aging scheme, where the priority of a process increases as it remains in the ready queue; hence, will eventually execute to completion
Here we are trying to develop programs for the above mentioned Scheduling techniques which would help understand the working of various algorithms better. Also we are going to conduct a comparative study of these Scheduling algorithms for better understanding the pros and cons of each algorithms and their needs and requirements.

II. LITERATURE SURVEY

A lot of work has already been done in this field by may research enthusiasts. Some of the important and interesting works which I came across while my study are mentioned here. Muhammad Umar Farooq et al [1] Developed an efficient Round Robin algorithm using Dynamic Time Quantum. Some such systems have already been developed but they take advantage of other algorithms and their running time is much higher due to sorting of processes which is practically impossible. So, our goal is to reduce running time of an algorithm along with efficiency constraints such as context switches, average waiting and turnaround times. In another work, A. Shah and K. Kotecha [2] proposed an ACO based scheduling algorithm for real-time operating systems (RTOS). During simulation, results were obtained with periodic tasks, measured in terms of Success Ratio & Effective CPU Utilization and compared with Earliest Deadline First (EDF) algorithm in the same environment. Sindhu M. et al [3] proposed an algorithm which can handle all types of process with optimum scheduling criteria. The intention of the OS should allow the process as many as possible running at all the time in order to make best use of CPU. In another similar work [4], The paper presents a method to customize the scheduling algorithm and the resource access protocol of an OSEK OS using aspect-oriented programming (AOP). We define aspects to replace the fixed priority scheduling mechanism of the OSEK OS with an Earliest Deadline First (EDF) scheduling mechanism or a Rate Monotonic Critical Laxity (RMCL) scheduling mechanism. A. Karapici et al in The Simulation of Round Robin and Priority Scheduling Algorithm [5] said that an operating system has some rules that control the execution of processes. These rules are called the scheduling algorithms. Scheduling algorithms make the organization and management of resources in a hardware. Two of these algorithms are round robin and priority scheduling. In the paper the authors stimulated the logic of these algorithms with two simple programs written in c language. In another paper, [6] the authors presented a dynamic scheduling algorithm that made use of genetic algorithm operations for scheduling multi non-pre-emptive task on uniprocessor system to get minimum average waiting time and average turnaround time. X. Yumei and C. Yimin in "An Improved Task Scheduling Algorithm in Embedded Systems," [7] focussed on analysing the task management and scheduling algorithm of embedded systems, and presented an improved embedded system scheduling algorithm and increase of time slice cycle algorithm. The embedded system was improved upon a real-time system, in which assignment priority scheduling is primary and time slice cycle scheduling is secondary. Xu Jiayi in his work [8] mentioned that the real-time scheduling algorithm of embedded Linux is especially discussed and an instance of priority-driven scheduling algorithm is illustrated in detail. In this example, the problem of priority inversion in process scheduling was solved perfectly. In another paper titled *A new Improved Round Robin-Based Scheduling Algorithm-A comparative Analysis* [9], the authors mentioned the paper to be three-fold: (1) presenting a survey of various RR based scheduling algorithms proposed and found in literature, (2) proposing a new RR based approach named, the Eighty-Five Percentile Round-Robin algorithm (EFPRR), to overcome the aforementioned issues, and (3) Conducting comparisons between eight RR based algorithms and the proposed approach using datasets with different characteristics. Extensive

experiments have been done to test the proposed approach. M. A. Shah et al in *" Organization Based Intelligent Process Scheduling Algorithm (OIPSA)"* [10] mentioned that the Organization Based Intelligent Process Scheduling Algorithm (OIPSA) intelligently learns the processes that are frequently used within an organization's operating system and give priority to the users' most wanted processes. In another related work titled *"A task scheduling algorithm Based on μC/OS-II system,"* [11], the authors mentioned that Embedded systems' Real-time and multi-mission capability largely depends on its task scheduling mechanism, based on the research of many existing real-time task scheduling algorithm and μC/OS-II's own characteristics, we proposed a new task scheduling algorithm which integrates dynamic priority and static priority and create a new scheduling mechanism. In another work *"A Real-Time Process Scheduling Policy in Windows,"[12],* the authors stated that the policy make use of Microsoft windows' process affinity and the clock interrupt technology. Affinity can limit the process or thread to work on a subset of core on the available CPU, and high-frequency clock interrupt service can drive the real-time processes or threads to switch at an appropriate time. Interrupt Service Routine, that is executed once every clock interrupt, is core of pre-emptive process scheduling algorithm. A similar work[13] also stated that the cause of priority inversion and the effect on system real-time performance, and two approaches named priority inheritance protocol and priority ceiling protocol are proposed to be used in uC/OS-II for restraining priority inversion phenomenon. Experiments testing show that the proposed subsystem hybrid scheduling algorithm and the method to realize restraining priority inversion protocols in uC/OS-II are feasible and effective. R. Srujana, Y. M. Roopa and M. D. S. K. Mohan, in *"Sorted Round Robin Algorithm,"* [17] Developed a new algorithm and stated that their algorithm was a combined product of the shortest job first (SJF) algorithm and Round Robin (RR) algorithm. It retained the advantage provided by these algorithms that may have an impact on the overall performance of the CPU and hence, is used to overcome the drawbacks in the RR algorithm by developing the strategies in use. S. Meng, Q. Zhu and F. Xia, in *"Improvement of the Dynamic Priority Scheduling Algorithm Based on a Heapsort,"* [18] used a heapsort algorithm with a lower time complexity to sort the comprehensive priority index so as to reduce the sorting overhead of the system. The system sorting overhead was then introduced to improve the decision condition of the priority scheduling subset and expand the schedulable range of the algorithm

III. PROPOSED FRAMEWORK

The project revolves around building models to depict the functioning of various task scheduling algorithms. The generated codes describe how the task scheduling algorithms schedule the tasks ready to be executed, for execution. Further, we propose a comparative study of these scheduling algorithms.
We intend to measure and compare the efficiency of these algorithms by means of metrices like the wait time, average waiting time, turnaround time, Completion time, response time. These metrices help compare and conclude the efficiency of each scheduling algorithm used.

IV. MODULE DESCRIPTION

In this project, attempt is being made to generate codes to depict the functionality of various task scheduling algorithms like First Come First Serve (FCFS), Shortest Job First (SJF), Shortest remaining Time First (SRTF), Round Robin (RR), Priority Scheduling. These algorithm codes are written in C language as the core operating system developers this language for development. The codes take various process numbers, Burst time, Arrival Time as well as priority of these processes as input. These codes generate the sequence in which these processes are executed by these algorithms, which help us understand the sequence of execution of processes in different algorithms. The code also outputs the metrices like the waiting time for each process, average wait time, turnaround time, etc which help us understand and compare the efficiency of these algorithms.
The programming language used for developing the codes is C/C++ as mostly all of the OS related codes are originally done in this language. Further, we prepared a set of test cases to feed to these codes, and see the output they generate. The output includes the sequence in which the input tasks get executed, The waiting time, completion time, turnaround time of each task given as input, as well as calculates and outputs the average waiting time and average turnaround time of each test case.
All these results recorded in an excel sheet to generate further insights from the data, and to perform comparison of the algorithms using certain graphs and other visualisation techniques.
The input to these algorithm codes include the list of process numbers, arrival time of each process, burst time of each process. For priority pre-emptive algorithm, the input also included the list of priorities to various tasks assigned.

V. RESULTS

The coded algorithm is tested against a variety of test cases including different number of processes ranging from seven processes to 20 processes as input to the algorithm. The algorithm produces desirable output including the completion time, waiting time, turnaround time of each process in tabular format as well as the average waiting time and the average turnaround time for the input set of processes as:

```
------------------FCFS----------------
Enter total no. of Processes: 10

Enter burst time and arrival time of each process:
P1: 15 0
P2: 7 1
P3: 3 2
P4: 6 3
P5: 5 4
P6: 10 15
P7: 1 5
P8: 8 6
P9: 12 7
P10: 20 10

Process_id  Burst_time  Completion_time  Turn_Around_time  Waiting_time
P1          15          15               15                0
P2          7           22               21                15
P3          3           25               23                22
P4          6           31               28                25
P5          5           36               32                31
P6          10          46               31                36
P7          1           47               42                46
P8          8           55               49                47
P9          12          67               60                55
P10         20          87               77                67

       Average waitig time is: 34.400002
       Average turn around time is: 37.799999
--------------------------------
Process exited after 74.47 seconds with return value 40
Press any key to continue . . .
```

Figure 1

Figure 1 depicts the output format designed for each algorithm code. In figure 1, the output is from the code of FCFS algorithm run using a set of 10 processes given as input. For testing the codes and to generate the required results, we have prepared a set of test cases to run on the codes and the results of each test case is recorded in an excel sheet so as to perform further comparison and visualization. We have recorded the average waiting time and average turnaround time of each test case in the following format:

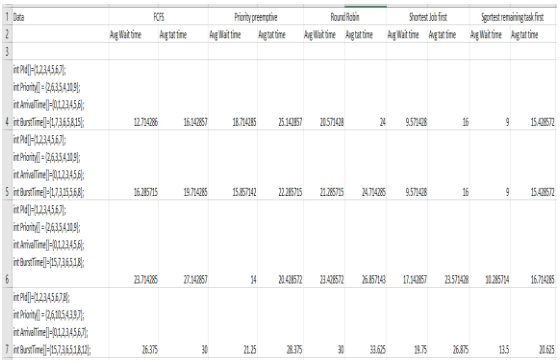

Figure 2

Further, we went one step ahead to record and analyse the change in the average waiting time and average turnaround time with the increase in the time quantum in Round Robin algorithm. For this, the code of Round Robin algorithm was run on the same input case but for different values of time quantum ranging from 1 to 20. These results were also recorded in an excel sheet to form a dataset for further visualisations.

Round Robin Algorithm		
Time Quantum	**average waiting time**	**average turnaround time**
1	78.400002	80.066666
2	78.133331	79.800003
3	76.400002	78.066666
4	77.400002	79.066666
5	75.066666	76.73333
6	72.066666	73.73333
7	70.933334	72.599998
8	67.26667	68.933334
9	70.066666	71.73333
10	63.066666	64.73333
11	64.599998	66.26667
12	64.599998	66.26667
13	66.066666	67.73333
14	67.533333	69.199997
15	61.599998	63.266666
16	62.133335	63.799999
17	62.666668	64.333336
18	63.200001	64.866669
19	63.733334	65.400002
20	59.666668	61.333332

Figure 3

The above two datasets are prepared by filling in the data from the output generated by different task scheduling algorithm codes for different inputs given. On this data, further visualisations is carried out in order to perform comparative analysis of various task scheduling algorithms.

VI. VISUALIZATION

The data recorded from the algorithm codes is used for visualising and comparing the functioning of various task scheduling algorithms in OS. Following are some important visualisations and inferences drawn from the data.

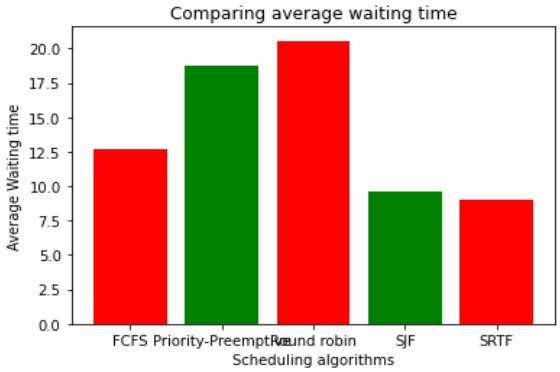

Figure 4

Figure 3 represents the average waiting time of various algorithms provided same input data to all algorithms. This implies that prioritizing the tasks can lead to a longer waiting time as in priority pre-emptive algorithm, since the small tasks with lower priority has to wait if the larger task with higher priority is getting executed. It can also be seen that the algorithms- SJF and SRTF give almost equal and lesser waiting time as compared to other algorithms, SRTF being the most efficient.

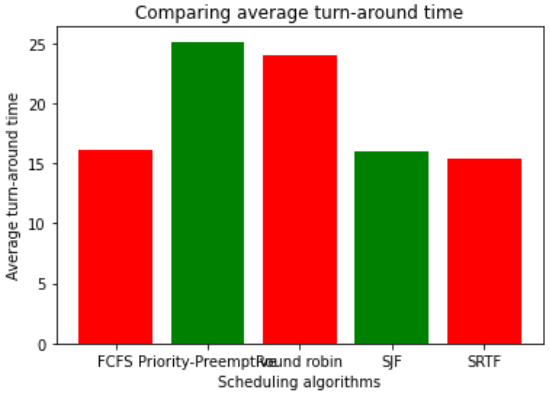

Figure 5

Figure 4 represents the average turnaround time of various algorithms. As can be seen, the priority pre-emptive scheduling algorithm gives the highest turnaround time because of the small tasks with lower priority having to wait for the larger tasks with higher priority which in turn increases the overall throughput time. Here also the SRTF algorithm is seen to give the best functionality over all other algorithms.

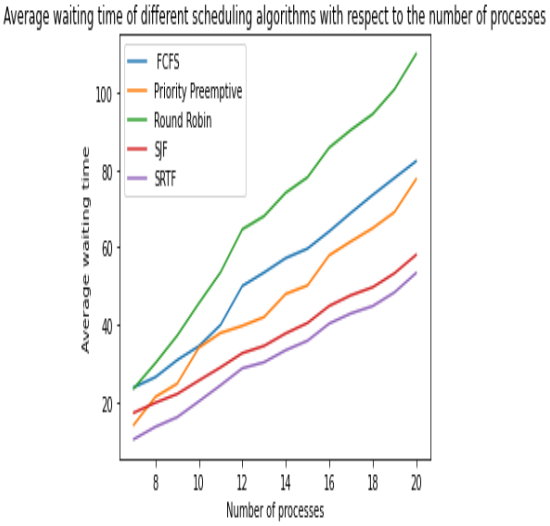

Figure 6

Figure 5 represents the plot of five different algorithms. Each line corresponds to a different algorithm. On the x axis is the number of processes given as input in each test case and the corresponding y axis represents the average waiting time for each test case generated by each algorithm. Thus the plot gives an idea about how the average waiting time increases with the increase in the number of input processes. From the graph it can be clearly inferred that with the increase in the number of input processes, the average waiting time for the round robin algorithm increases the most and very rapidly. Also, we can see that the algorithm whose average waiting time is least for every number of processes is SRTF algorithm.

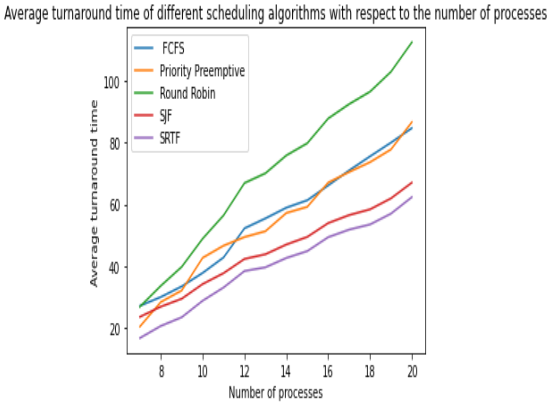

Figure 7

Figure 6 depicts the plot of the change in the average turnaround time of each algorithm wrt to the increase in the number of input processes. The plot again says that with the increase in the number of input processes, round robin algorithm remains with the highest turnaround time while SRTF algorithm remains with the lowest turnaround time all throughout, for each value of the number of input processes.

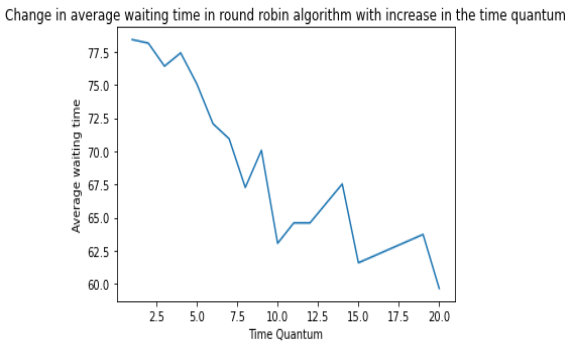

Figure 8

Figure 7 depicts the plot between the average waiting time and the different values of time quantum. The graph depicts the change in the average waiting time, when same set of input processes is given to the Round Robin algorithm but

for different amounts of time quantum. From the above plot it can be seen that with the increase in the amount of time quantum, the average waiting time for the algorithm is quite irregular. The average waiting time is seen to be increasing at points for certain value of time quantum and then deceasing again. On the whole the average waiting time for Round Robin algorithm is seen to gradually decrease with the increase in the values of time quantum.

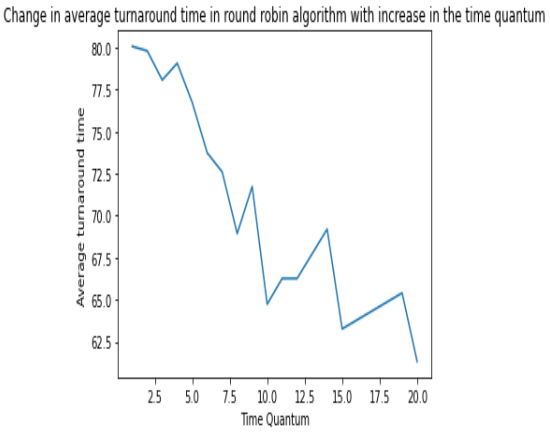

Figure 9

In Figure 8, the plot shows the change in the average turnaround time for same set of processes as input but for different amounts of time quantum. As the plot for the average waiting time, the plot for the average turnaround time also shows an irregular behaviour, yet is seen to decrease with the increase in the amount of the time quantum.

VII. DISCUSSIONS

After analysing the results from the codes of different algorithms, comparing the results and visualising the results, we can draw some important inferences. We can conclude that the priority pre-emptive algorithm may be appropriate for priority based tasks, but proves to be quite inefficient when used in other cases. This is because the heavy tasks with higher priority may be provided with the CPU resources and the smaller tasks having lower priority may be waiting for a long time for their turn. A few enhancements to the basic priority pre-emptive algorithm are also introduced in which the waiting tasks are promoted to a higher priority after each cycle of execution, which solves the problem of long term starvation. Secondly, we can see that the SRTF functions the best among all other algorithms under consideratuin. Lastly it is also seen that the average waiting time and average turnaround time of the Round Robin algorithm gradually decreases with the increase in the amount of time quantum.

VIII. REFERENCES

[1] M. U. Farooq, A. Shakoor and A. B. Siddique, "An Efficient Dynamic Round Robin algorithm for CPU scheduling," 2017 International Conference on Communication, Computing and Digital Systems (C-CODE), Islamabad, 2017, pp. 244-248.

[2] A. Shah and K. Kotecha, "Scheduling Algorithm for Real-Time Operating Systems Using ACO," 2010 International Conference on Computational Intelligence and Communication Networks, Bhopal, 2010, pp. 617-621.

[3] S. M., R. R. and V. P., "An Optimum Multilevel CPU Scheduling Algorithm," 2010 International Conference on Advances in Computer Engineering, Bangalore, 2010, pp. 90-94.

[4] Y. Harada, K. Abe, M. Yoo and T. Yokoyama, "Aspect-Oriented Customization of the Scheduling Algorithms and the Resource Access Protocols of a Real-Time Operating System Family," 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), Chengdu, 2015, pp. 87-94

[5] A. Karapici, E. Feka, I. Tafa and A. Allkoçi, "The Simulation of Round Robin and Priority Scheduling Algorithm," 2015 12th International Conference on Information Technology - New Generations, Las Vegas, NV, 2015, pp. 758-758.

[6] Q. M. Hussein and A. N. Hasoon, "Dynamic process scheduling using genetic algorithm," 2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT), Baghdad, 2017, pp. 111-115.

[7] X. Yumei and C. Yimin, "An Improved Task Scheduling Algorithm in Embedded Systems," 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, Jinggangshan, 2010, pp. 682-685.

[8] Xu Jiayi, Yan Chaojun, Wu Aiping and Zhou Qili, "Process scheduling of Linux in

embedded system," 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, 2010, pp. V11-25-V11-26.

[9] Y. H. Jbara, "A new Improved Round Robin-Based Scheduling Algorithm-A comparative Analysis," 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 2019, pp. 1-6.

[10] M. A. Shah, M. B. Shahid, S. Zhang, S. Mustafa and M. Hussain, "Organization Based Intelligent Process Scheduling Algorithm (OIPSA)," 2015 21st International Conference on Automation and Computing (ICAC), Glasgow, 2015, pp. 1-6.

[11] Shaowei Liu, Ying Bao and Hongbo Guan, "A task scheduling algorithm Based on μC/OS-II system," 2011 International Conference on Multimedia Technology, Hangzhou, 2011, pp. 4798-4800.

[12] L. Wang, X. S. Zhou, Z. J. Jiang and A. Zhang, "A Real-Time Process Scheduling Policy in Windows," 2012 International Conference on Computer Science and Service System, Nanjing, 2012, pp. 22-24

[13] X. Wang and T. Liu, "Research on Subsystem Hybrid Scheduling and Priority Inversion Based uCOS-II," 2012 Fifth International Conference on Intelligent Networks and Intelligent Systems, Tianjin, 2012, pp. 117-121.

[14] R. Srujana, Y. M. Roopa and M. D. S. K. Mohan, "Sorted Round Robin Algorithm," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2019, pp. 968-971.

[15] S. Meng, Q. Zhu and F. Xia, "Improvement of the Dynamic Priority Scheduling Algorithm Based on a Heapsort," in IEEE Access, vol. 7, pp. 68503-68510, 2019.