Scheduling Algorithms Comparisons and Analysis using Simulation Results

AL REFAI MOHAMMED N.
Software Engineering Department,
Zarqa University,
Zarqa,
JORDAN
ORCiD: https://orcid.org/0000-0002-3464-8900

Abstract: - Heuristics Process scheduling is one of the key components of any operating system design. It defines the way the CPU resources are shared among processes in the system. In this paper, we present the performance comparison of five different process scheduling algorithms: FIFO, Round Robin, Shortest Remaining Time, Shortest Processing Time, and Highest Response Ratio Next. We evaluated these algorithms based on the simulations and calculated their effect on three key metrics: turnaround time, waiting time, and CPU utilization. The results showed that each of them has its pros and cons, thus making the optimal choice of scheduling algorithm impossible. This paper can be used by operating system designers or professionals in selecting an appropriate scheduling algorithm for a certain system and learning objectives.

Key-Words: - Scheduling Algorithms, Operating System, Optimization, Process Scheduling, CPU utilization turnaround time, waiting time.

Received: December 13, 2023. Revised: July 14, 2024. Accepted: August 15, 2024. Published: September 20, 2024.

1 Introduction

Today, in modern computing systems, effective use of system resources is essential for the smooth running and high performance of the system. Process scheduling is one of the critical aspects of resource management since it relates to the order in which processes are allocated to the central processing unit, [1] . To begin with, the scheduling technique widely applied in different domains including the Internet of Things (IOT) used the scheduling method GRK for LBCs to provide the right trade-off between security, performance, and cos, [2]. Furthermore, the form also used scheduling distributed with generators and storage in customer-oriented economic benefits to prolong battery life, [3]. Other significant applications include load scheduling for isolated power systems, [4], mobile ad hoc networks, [5] , and exam scheduling challenges, [6]. Managing resource-constrained project scheduling problems, [7], efficiently utilizing cellular user resources to enhance network spectrum efficiency, [8], conducting cloud service analysis for quality-of-service aware task placement services, [9], [10], [11] on parallel machines, [12] and optimizing scheduling in industrial railway junctions, [13]. Several different algorithms can be used for process scheduling [14], each with its advantages and disadvantages. One of the simplest scheduling algorithms is FIFO, it was developed by the operating system's community as a basic algorithm in process scheduling. Round Robin, is a more advanced algorithm that allows each process to use the CPU for a certain time (quantum) before being preempted, it was developed by IBM in the 1960s, [15]. Shortest Remaining Time (SRT) is a scheduling algorithm that prioritizes the process with the shortest remaining time, [16]. Although proposed Shortest Processing Time (SPT) to schedule the process with the shortest burst time, [17] . As reported in [18] developed HRRN to schedule the process with the highest ratio of (Waiting time + burst time) / burst time [18]. From the list of well-known scheduling above, the paper primarily distinguishes the work performance among the five scheduling algorithms: First In First Out FIFO, Round Robin, Shortest Remaining Time SRT, Shortest Processing Time SPT, and Highest response ratio next HRRN.

The selection of these algorithms is attributed to their popularity and having been proposed as potential, alternatives to what previous research used the traditional algorithms. The simulation of the algorithms while determining metrics such as the turnaround time, waiting time, and CPU utilization

is critical in developing insightful information regarding the variation in the algorithms and the effect they have on the multiprogramming system's performance. Moreover, apart from merely comparing the performance of the scheduling algorithms, this research paper will consider the burst time of the processes and their random arrival times, Also the simulation considers that the time quantum for every scheduling set of rules can be 10. Specifically, the burst time of each system was randomly generated among 15 to 55 units of time, and the arrival time of every process can also be randomly generated. By simulating the scheduling algorithms under these conditions, we hope to benefit from a more practical knowledge of ways they are carried out in an actual scenario where processes arrive at unpredictable intervals and have varying burst times with a time quantum of 20. This aspect could be a key consideration within the simulation and analysis of the performance of these scheduling algorithms.

Overall, this research pursuits to offer a complete evaluation of the different scheduling algorithms and assist in becoming aware of the pleasant algorithm for a given system requirement. Average waiting time measures the time a procedure spends ready in the prepared queue before it's far executed. Lower average waiting times imply better system performance, as approaches don't spend a whole lot of time waiting for the CPU, [19], [20], [21]. Average turnaround time measures the total time from a process arrives in the system until it completes execution and departs the system. Lower average turnaround times indicate better system performance, as processes are completing execution more quickly, [22]. Average response time measures the total time from a process submitted to the system until it first receives a response from the CPU, [23], [24]. Lower average response times indicate better system performance, as processes are receiving a response more quickly, [25], [26]. CPU utilization measures the percentage of time the CPU is busy executing processes. Higher CPU utilization indicates better system performance, as the CPU is being used more effectively, [15], [27]. The rest of this research, explains the research methodology in section two, the Implementation in section three, the results and comparisons in sections four and five consequently, and the conclusion in section six.

2 Research Methodology

This research works on all the primary scheduling algorithms to compare performance metrics and extract differences: The first step was reading the previous work related to scheduling algorithms to have the required knowledge as written in the introduction section. Then create a simulation using Java programming language for all primary scheduling algorithms in the second step. After that select three datasets with 100,1000, and 10000 processes consequently, with arrival time and burst time for each process. In the fourth step obtain the performance metrics outcomes for each scheduling technique. In the last step compare the performance metrics results, and then provide an explanation, summary, and recommendations based on the results.

3 Implementation

This research compares the performance of five scheduling algorithms (FIFO, Round Robin, SRT, SPT, and HRRN) by simulating and measuring all performance metrics (turnaround time, waiting time, Response Time, and CPU utilization) then makes a comprehensive comparison to extract the results and finds.

3.1 First In First Out (FIFO)

Also known as First Come First Served, is a scheduling algorithm that executes processes in the order they arrive in the ready queue. This means that the process that has been waiting for the longest is executed first. FIFO is a simple algorithm and easy to implement, making it a popular choice for systems with basic scheduling requirements, [28].

One advantage of FIFO is its equity, as all methods are treated similarly, and given the equal quantity of CPU time. It is predictable, as the order in which techniques are completed is decided via the order wherein they come within the ready queue. On the other hand, FIFO can bring about lengthy waiting times for strategies, in particular when there are many approaches inside the queue or some procedures have longer runtimes. This can result in negative device performance and low CPU utilization, [29].

Additionally, FIFO is not appropriate for real-time structures because it does not bear in mind the timing constraints of processes. To compare the performance of the FIFO algorithm, we conducted simulations with the use of units of a hundred, a thousand, and ten thousand processes with various arrival times and run instances. The consequences of the simulations are shown in Table 1.

Table 1. Simulation Results for FIFO

Metric	100 processes	1000 processes	10000 processes
Turnaround time	28.95	29.49	29.78
Waiting time	10.78	11.35	11.81
CPU utilization	0.68	0.67	0.66
Response time	10.78	11.35	11.81

3.2 Round Robin

Round Robin is a scheduling algorithm that permits each process to run for a hard and fast amount of time, referred to as the time slice, before being preempted and placed on the top of the ready queue, [28] . This approach ensures that each process gets a truthful proportion of the CPU and prevents any single technique from monopolizing the resource. The time slice is typically set to a small value, along with 20 or 30 units of time, to make sure that approaches are preempted often, [1], [30]. One advantage of Round Robin is that it's far more honest, as all strategies are given the same get entry to the CPU. It is also suitable for actual-time systems, as it can aid strategies with strict timing constraints by means of placing the time slice to a small value, [31] . However, spherical-robin has a few negative aspects as well, one disadvantage is that it can suffer from overhead, as the ready queue must be constantly updated. Processes must be preempted and context switched frequently. Round Robin is also not efficient for long-running processes, as they may have, [19], [30], [32]. To evaluate the performance of the Round Robin algorithm, we conducted simulations using the same data sets of 100, 1000, and 10000 processes with a time slice (quantum) of 5 units of time. The results of the simulations are shown in Table 2.

Table 2. Simulation Results for Round Robin

Metric	100 processes	1000 processes	10000 processes
Turnaround time	37.56	37.23	37.087
Waiting time	19.39	19.1	19.12
CPU utilization	0.68	0.67	0.66
Response time	2.45	2.65	2.57

3.3 Shortest Remaining Time (SRT)

Shortest Remaining Time (SRT) is a scheduling algorithm that favors processes with the shortest remaining run time, [30]. When a process becomes ready, it is executed if it has the shortest remaining run time of all the processes in the ready queue. If a process is preempted before it completes, it is placed at the front of the ready queue, as it now has the shortest remaining run time.

One gain of SRT is that it is able to enhance machine performance with the aid of lowering waiting time and turnaround time for processes, [33]. It is also suitable for real-time structures, as it can prioritize methods with strict timing constraints. However, SRT has some dangers as properly. One drawback is that it may suffer from overhead, as the ready queue should be continuously taken care of primarily based on the remaining run times of the processes, [34]. SRT is also not fair, as some procedures may be starved of CPU time if they have longer run time. To examine the performance of the SRT algorithm, we performed simulations with the use of sets of a hundred, a thousand, and ten thousand processes with varying arrival times and run times. The outcomes of the simulations are shown in Table 3.

Table 3. Simulation Results for SRT

Metric	100 processes	1000 processes	10000 processes
Turnaround time	26.14	26.241	26.04
Waiting time	7.97	8.103	8.07
CPU utilization	0.68	0.67	0.66
Response time	4.1	3.277	4.18

3.4 Shortest Processing Time (SPT)

Shortest Processing Time (SPT) is a scheduling algorithm that favors processes with the shortest run time, [17]. When a process becomes ready, it is executed if it has the shortest run time of all the processes in the ready queue.

One advantage of SPT is that it can improve system performance by reducing waiting times and turnaround times for processes, [35]. It is also fair, as all processes are given an equal chance to access the CPU based on their run times. However, SPT has some disadvantages as well.

One disadvantage is that it can suffer from overhead, as the ready queue must be constantly sorted based on the run times of the processes. SPT is also not suitable for real-time systems, as it does not take into account the timing constraints of processes. To evaluate the performance of the SPT algorithm, we conducted simulations using sets of 100, 1000, and 10000 processes with varying arrival times and run times. The results of the simulations are shown in Table 4.

One disadvantage is that it may be afflicted by overhead because the equipped queue ought to be continuously looked after primarily based on the run times of the processes, [18]. SPT is likewise not appropriate for real-time structures, as it no longer takes into account the timing constraints of techniques. To evaluate the overall performance of the SPT algorithm, we performed simulations with the use of units of one hundred, one thousand, and ten thousand processes with varying arrival times and run instances. The consequences of the simulations are proven in Table 4.

Table 4. Simulation Results for SPT

Metric	100 processes	1000 processes	10000 processes
Turnaround time	27.41	27.698	27.48
Waiting time	9.24	9.56	9.51
CPU utilization	0.68	0.67	0.66
Response time	9.24	9.56	9.51

3.5 Highest Response Ratio Next (HRRN)

Highest Response Ratio Next (HRRN) is a scheduling algorithm that favors strategies with the highest response time, [22]. The response ratio of a process is calculated as the ratio of the waiting time to the run time of the process. When the process turns ready, it is performed if it has the best response ratio of all of the procedures in the ready queue. One benefit of HRRN is that it is able to improve system performance by decreasing waiting times and turnaround times for processes, [35]. It is also truthful, as all processes are given an equal chance to get the right of entry to the CPU primarily based on their response ratio.

However, HRRN has some disadvantages as well. One disadvantage is that it can suffer from overhead, as the ready queue must be constantly sorted based on the response ratios of the processes, [36]. HRRN is also not suitable for real-time systems, as it does not take into account the timing constraints of processes. To evaluate the performance of the HRRN algorithm, we conducted simulations using sets of 100, 1000, and 10000 processes with varying arrival times and run times. The results of the simulations are shown in Table 5.

In all simulations, the CPU utilization was measured and found to be 0.69 for the 100 process simulation, 0.67 for the 1000 process simulation, and 0.66 for the 10000 process simulation. This indicates that the CPU was being used for 69%, 67%, and 66% of the total time respectively. It's important to note that the utilization may vary depending on the number of processes and the specific scheduling algorithm being used, [37].

Table 5. Simulation Results for HRRN

Metric	100 processes	1000 processes	10000 processes
Turnaround time	28.3	28.566	28.522
Waiting time	10.13	10.428	10.55
CPU utilization	0.68	0.67	0.66
Response time	10.13	10.428	10.55

4 Main Results

4.1 Average Turnaround Time

Based on the simulation results (Table 1, Table 2, Table 3, Table 4 and Table 5), the following figure shows the differences in the average turnaround time for all strategies using the 100, 1000, and 10000 processes:

From Figure 1 the following comparing notes were recorded:

a. SRT, SPT, and HRRN had the lowest average turnaround times among the five algorithms for all three numbers of processes (100, 1000, and 10000). These algorithms consistently had shorter average turnaround times than Round Robin and FIFO.
b. Round Robin had the highest average turnaround times among the five algorithms for all three numbers of processes. It had significantly longer average turnaround times than SRT, SPT, and HRRN, and slightly longer average turnaround times than FIFO.
c. FIFO had intermediate performance among the five algorithms for all three numbers of processes. It had lower average turnaround times than Round Robin but higher average turnaround times than SRT, SPT, and HRRN.

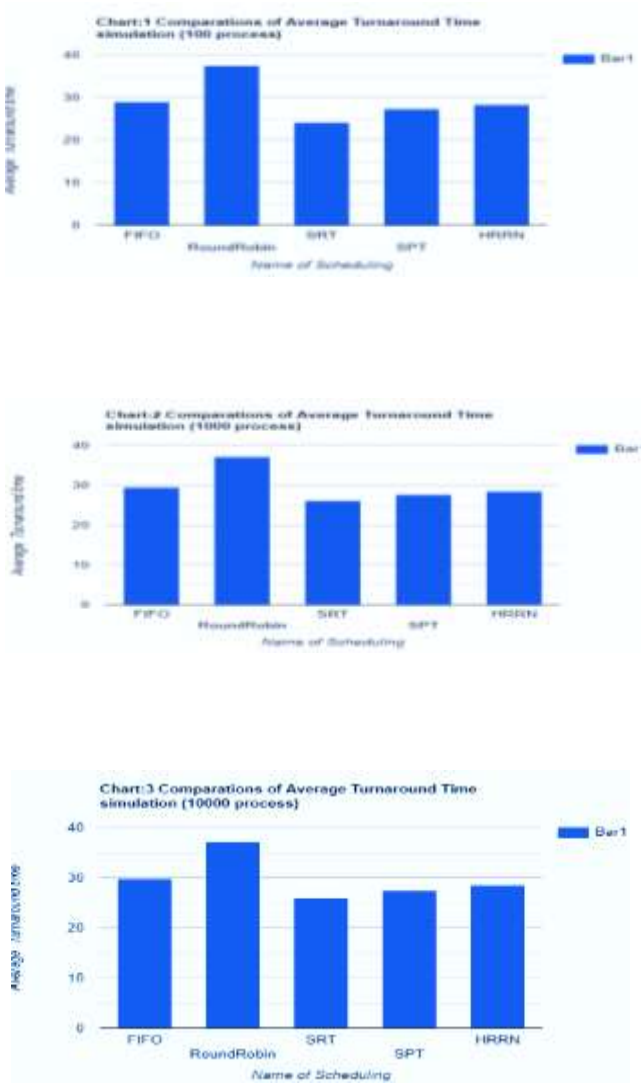

Fig. 1 Average turnaround time comparisons with 100, 1000, 10000 Process for all strategies

4.2 Average Waiting Time

Based on the simulation results and the bar charts for average waiting time Figure 2, we can note the following conclusions:

a. SRT, SPT, and HRRN had the lowest average waiting times among the five algorithms for all three numbers of processes (100, 1000, and 10000). These algorithms consistently had shorter average waiting times than FIFO and Round Robin, with differences of up to [8 Units of Time] for 100 processes, [8 Units of Time] for 1000 processes, and [8 Units of Time] for 10000 processes.

b. Round Robin had the highest average waiting times among the five algorithms for all three numbers of processes. It had significantly longer average waiting times than SRT, SPT, and HRRN, with differences of up to [11 Units of Time] for 100 processes, [11 Units of Time] for 1000 processes, and [11 Units of Time] for 10000 processes.

c. FIFO had intermediate performance among the five algorithms for all three numbers of processes. It had higher average waiting times than SRT, SPT, and HRRN, with differences of up to [2 Units of Time] for 100 processes, [2 Units of Time] for 1000 processes, and [2 Units of Time] for 10000 processes.

2.1 Average Response Time:

Based on the simulation results and the bar charts for average response time (Tables 7-9), we can draw the following conclusions from Figure 3:

a. In this scenario, it was observed that RR and SRT had the lowest average response times among the five algorithms for different numbers of processes (100, 1000, and 10000). These algorithms consistently had shorter average response times than HRRN, SPT, and FIFO.
It's crucial to observe that the lower average response time of the Round Robin and SRT algorithms as

b. compared to FIFO, SPT, and HRRN, is because of the truth that each Round Robin and SRT are preemptive algorithms. This manner that can interrupt a running process and flow on to the subsequent process, while FIFO, SPT, and HRRN are non-preemptive, which means they anticipate a procedure to finish before shifting on to the subsequent one. This lets Round Robin and SRT respond quickly to the new processes, which improves their response time.

c. Additionally, this also explains why they have got different response times in comparison to their waiting time, as they are able to move on to other processes and retain executing them, reducing response times, [36].

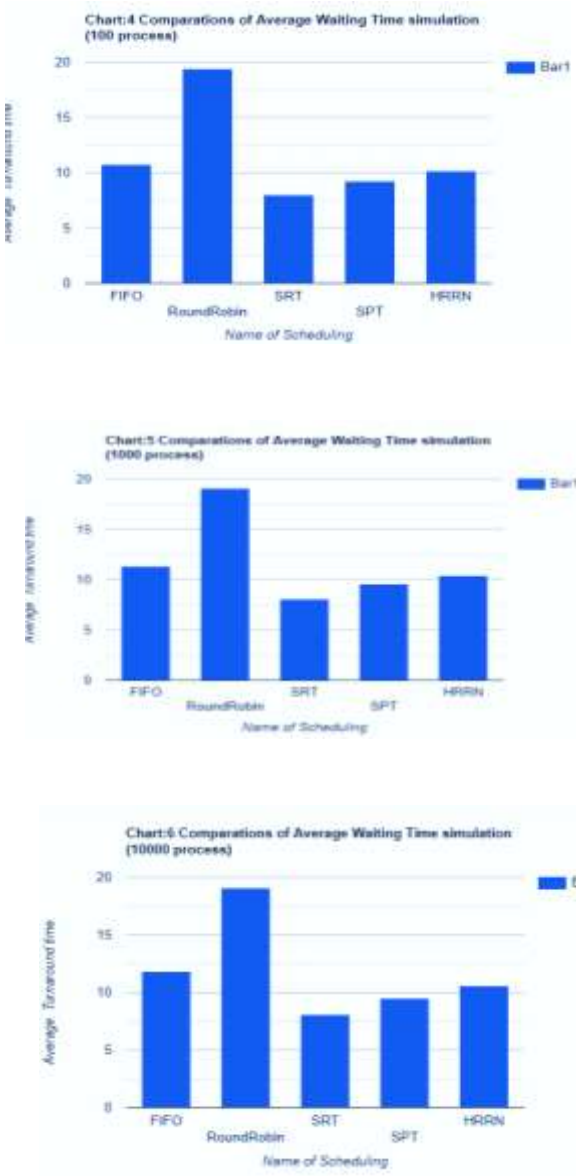

Fig. 2: Average waiting time comparisons with 100, 1000, 10000 Process for all strategies

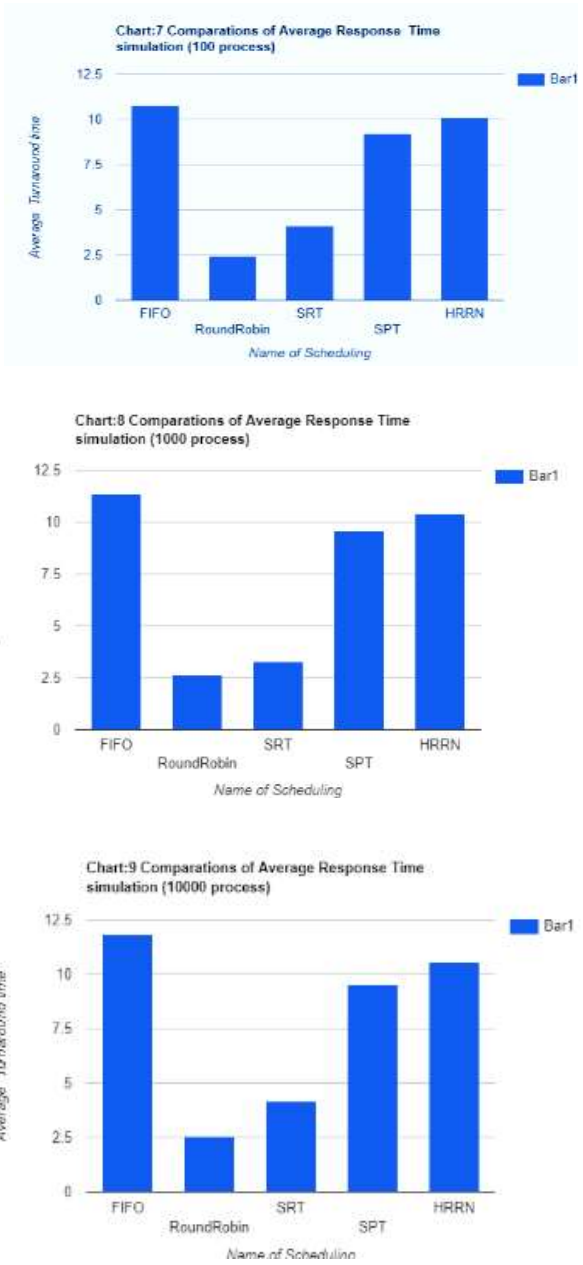

Fig. 3 Average response time comparisons with 100, 1000, 10000 Process for all strategies

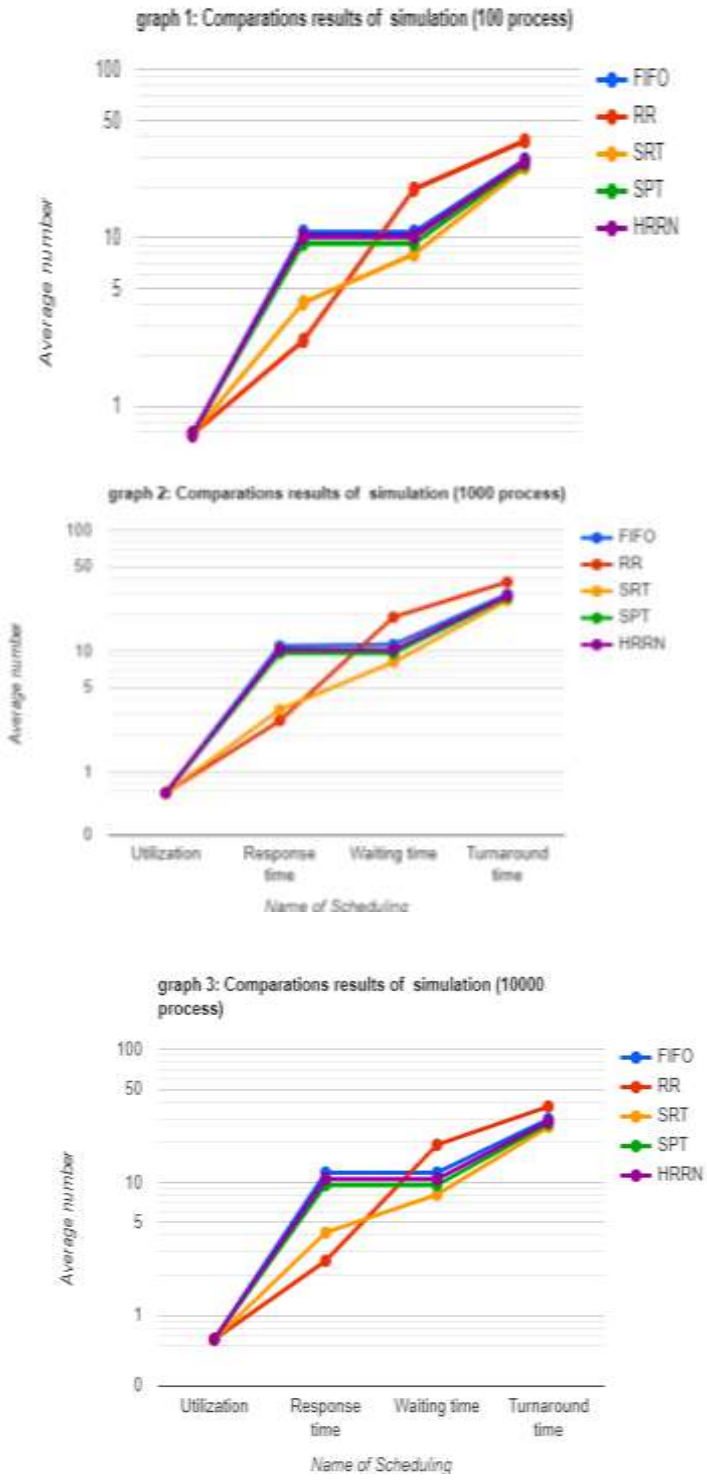

Fig. 4 show the comparision of the average waiting time, average turnaround time, response time, and utilization of the five algorithms, using data sets of 100,

5 Utilization

The CPU utilization results in our simulation were consistent across all three process numbers (100, 1000, and 10000) and for all five scheduling algorithms. This is likely because we used the same dataset for each simulation and did not take into account any additional factors such as multi-processor technology [24]. It's important to note that in a real-world scenario, other factors such as the number of processors, the number of active processes, and the system load may also affect CPU utilization. [15]

6 Comparisons Results

Figure 4 shows the comparison of the average waiting time, average turnaround time, response time, and utilization of the five algorithms, using data sets of 100, 1000, and 10000. The comparison is shown from the perspective of the four metrics mentioned.

7 Conclusion

Process scheduling algorithms are used to determine which system needs to be run by the OS at any given time. There are numerous distinctive

scheduling algorithms that might be usually used, consisting of FIFO, Round Robin, SRT, SPT, and HRRN. Each algorithm has its own exchange-offs and is better suited to specific conditions.

The performance of each scheduling algorithm can range relying on the unique characteristics of the processes being scheduled and the workload of the system. It is vital to cautiously compare the performance of various algorithms and pick the only one that high-quality meets the desires of your device.

It is also crucial to note that the results of a simulation may not generalize to all feasible scenarios. The overall performance of each scheduling algorithm may also change with one-of-a-kind datasets or beneath distinct situations. Therefore, it is vital to cautiously recollect the unique requirements and goals of your system whilst selecting a scheduling algorithm.

Acknowledgment:
This research is funded by the Deanship of Research and Graduate Studies in Zarqa University /Jordan. The authors also gratefully acknowledge the helpful

comments and suggestions of the reviewers, which have improved the research.

References:

[1] L. Datta, "Efficient Round Robin Scheduling Algorithm with Dynamic Time Slice," *International Journal of Education and Management Engineering (IJEME),* vol. Vol. 5, no. No. 2, p. DOI:10.5815/ijeme.2015.02.02, 8 Jun 2015.

[2] N. Kapalova, K. Algazy, A. Haumen and . K. Sakan, "Statistical analysis of the key scheduling of the new lightweight block cipher," *International Journal of Electrical and Computer Engineering,* vol. 13, no. 6, pp. 6817-6826,DOI:10.11591/ijece.v13i6.pp6817-6826, Decmnber 2023.

[3] N. Guru, S. Patnaik, M. Nayak and M. Viswavandya, "Wind generator and storage system scheduling for customer benefit and battery life," *Bulletin of Electrical Engineering and Informatics,* vol. 12, no. 5, pp. 2586-2594, DOI: 10.11591/eei.v12i5.4661, October 2023.

[4] V. Joy, J. John and S. Krishnakumar, "Backpropagation neural network based adaptive load scheduling method in an isolated power system," *Bulletin of Electrical Engineering and Informatics,* vol. 12, no. 5, pp. 3000-3007, DOI: 10.11591/eei.v12i5.4511, October 2023.

[5] G. Rengarajan, . N. Ramalingam and K. Suriyan, "Performance enhancement of mobile ad hoc network life time using energy efficient techniques," *Bulletin of Electrical Engineering and Informatics,* vol. 12, no. 5, pp. 2870-2877, DOI:10.11591/eei.v12i5.5184, October 2023.

[6] T. Alrawashdeh, . K. Al-Moghrabi and A. Al-Ghonmein, "A profiling-based algorithm for exams' scheduling problem," *International Journal of Electrical and Computer Engineering,* vol. 13, no. 5, pp. 5483-5490, DOI:10.11591/ijece.v13i5.pp5483-5490, October 2023.

[7] B. Roy and A. Sen, "An integrated hybrid metaheuristic model for the constrained scheduling problem," *IAES International Journal of Artificial Intelligence,* vol. 12, no. 3, pp. 1083-1095,DOI: 10.11591/i.v12.i3, September 2023.

[8] D. Kesavan, E. Periathambi and A. Chockkalingam, "bipartite graph based proportional fair scheduling strategy to improve throughput with multiple resource blocks," *International Journal of Electrical and Computer Engineering,* vol. 13, no. 4, pp. 4280-4290, DOI: 10.11591/ijece.v13i4.pp4280-4290, August 2023.

[9] N. Pakhrudin, M. Kassim and . A. Idris, "Cloud service analysis using round-robin algorithm for quality-of-service aware task placement for internet of things services," *International Journal of Electrical and Computer Engineering,* vol. 13, no. 3, pp. 3464-3473, DOI: 10.11591/ijece.v13i3.pp3464-3473, June 2023.

[10] N. George, A. Kadan and . V. Vijayan, "Multi-objective load balancing in cloud infrastructure through fuzzy based decision making and genetic algorithm based optimization," *IAES International Journal of Artificial Intelligence,* vol. 12, no. 2, pp. 678-685, DOI: 10.11591/ijai.v12.i2.pp678-685, June 2023.

[11] N. Ghazy, A. Abdelkader, M. Zaki and K. Eldahshan, "An ameliorated Round Robin algorithm in the cloud computing for task scheduling," *Bulletin of Electrical Engineering and Informatics,* vol. 12, no. 2, pp. 1103-1114, DOI:10.11591/eei.v12i2.4524, April 2023.

[12] . M. Kahar and . M. Abed, "Hybridizing genetic algorithm and single-based metaheuristics to solve unrelated parallel machine scheduling problem with scarce resources," *IAES International Journal of Artificial Intelligence,* vol. 12, no. 1, pp. 315-327,DOI: 10.11591/ijai.v12.i1.pp315-327, March 2023.

[13] L. Bogdanova, S. Nagibin, . D. Loskutov and N. Goncharova , "Neuro-fuzzy-based mathematical model of dispatching of an industrial railway junction," *Bulletin of Electrical Engineering and Informatics,* vol. 12, no. 1, pp. 502-513, DOI: 10.11591/eei.v12i1.4055, February 2023.

[14] M. Kumar, S. Sharma, A. Goel and S.P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications,* vol. 143, no. 1, pp. 1-33 , DOI:10.1016/j.jnca.2019.06.006., January 2019.

[15] Ubita and M. Sharama, "Comparison of Various CPU Scheduling," *International Journal of Creative Research Thoughts (IJCRT),* vol. 10, no. 6, pp. ISSN: 2320-2882, 2022.

[16] W. J. Layland and . L. L. Chang, "A Taxonomy

of Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM,* vol. 20, no. 1, pp. 46–61, DOI:10.1145/321738.321743, 1 Jan 1973.

[17] A. E. Evwiekpaefe, A. Ibrahim and M. N. Musa, ""Improved Shortest Job First CPU Scheduling Algorithm"," *Dutse Journal of Pure and Applied Sciences (DUJOPAS),* vol. 8, no. 3, pp. 115-127, September 2022.

[18] J. J. Sharma and A. K. Pandey, "A Highest Response Ratio Next(HRRN) Algorithm Based Load Balancing Policy For Cloud Computing.," *International Journal of Computer Science Trends and Technology (IJCST),* vol. 3, no. 1, pp. 72-76, https://www.ijcstjournal.org/volume-3/issue-1/IJCST-V3I1P14.pdf ,accessed 15 JUL 2024, Jan-Feb 2015.

[19] G. Tomsho, Guide to Operating Systems, 5th edition ed., Cengage Learning, August 30, 2016.

[20] W. Stallings , Operating Systems: Internals and Design Principles, 8th edtion, ISBN: 978-0134670959 ed., 2. March 13, Ed., Pearson, 2017.

[21] D. Biswas, M. Samsuddoha, M. R. A. Asif and M. M. Ahmed, "Optimized Round Robin Scheduling Algorithm Using Dynamic Time Quantum Approach in Cloud Computing Environment," *International Journal of Intelligent Systems and Applications(IJISA),* vol. 15, no. 1, pp. 22-34, DOI:10.5815/ijisa.2023.01.03, February 2023.

[22] A. Silberschatz, P. B. Galvin and G. Gagne , Operating system concepts., 10th ed., Wiley & sons, 2018.

[23] D. Biswas and M. Samsuddoha, "Determining Proficient Time Quantum to Improve the Performance of Round Robin Scheduling Algorithm," *International Journal of Modern Education and Computer Science(IJMECS),* vol. 8, no. 10, pp. 33-40, 2019. DOI:10.5815/ijmecs.2019.10.04, 2019.

[24] Y. Wiseman and S. Jiang, Advanced operating systems and kernel applications, 1st ed., New York: Information Science Reference, 2009, pp. ISBN-10 : 1605668508.

[25] M. Akhtar, B. Hamid, I. ur-Rehman, M. Humayun, M. Hamayun and H. Khurshid, "An Optimized Shortest job first Scheduling Algorithm for CPU Scheduling," *J. Appl. Environ. Biol. Sci.,* vol. 12, no. 5, pp. 42-46, ISSN: 2090-4274 , 2015.

[26] J. Shivani and M. Manisha, "AFTM-Agent Based Fault Tolerance Manager in Cloud Environment," *The International Arab Journal of Information Technology,* vol. 19, no. 3, pp. 306-402 , DOI: 10.34028/iajit/19/3/14, 2022.

[27] P. Chen, H. Chen, W. Liu, L. Long, W. Chang and a. N. Guan, "DAG-Order: An Order-Based Dynamic DAG Scheduling for Real-Time Networks-on-Chip," *ACM Transactions on Architecture and Code Optimization,* vol. Vol. 21, no. No. 1, pp. 1–24, DOI:10.1145/3631527, 2023.

[28] W. S.-N. Sarayut Phornchroen, "A proposed Round Robin Scheduling Algorithmfor Enhancing Performance of CPU Utilization," *Przeglad Elektotechniczny,* vol. 4, no. 4, pp. 2083-2097, , 4 2018.

[29] G. Martin and D. E. Estrin, "Models of Computational Systems-Cyclic to Acyclic Graph Transformations," *IEEE Transactions on Electronic Computers,* Vols. EC-16, no. no. 1, pp. 70-79 , DOI: 10.1109/PGEC.1967.264607, February 1967.

[30] K. A. PATEL, "Analysis and Testing of Different Time Quantum for Round Robin Algorithm," *International Journal of Advances in Electronics and Computer Science,* vol. 3, no. 4, pp. 40-44, ISSN: 2393-2835, Apr 2016.

[31] P. Ramesh and U. Ramachandraiah, "Performance evaluation of real time scheduling algorithms for multiprocessor systems," in *International Conference on Robotics, Automation, Control and Embedded Systems (RACE)*, CHENNAI, India, 2015.

[32] S. Zouaoui, L. Boussaid and M. Abdellatif, "Priority based round robin (PBRR) CPU scheduling algorithm," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 9, no. 1, pp. 190-202, DOI: 10.11591/ijece.v9i1.pp190-202, 2019.

[33] A. P. Laplante and J. S. Ovaska, Real-Time Systems Design and Analysis, Tools for the Practitioner, 4th ed., John Wiley & sons INC, Publication, 2012, pp. ISBN: 978-0470768648, ISBN: 978-0470768648.

[34] H. B. Andrew S. Tanenbaum, Modern Operating System, vol. 5th, Pearson - India, 2016.

[35] B. P. Daniel and M. Cesati, Understanding the Linux Kernel, THIRD EDITION ed., O'Reilly, 2005, pp. 0-944.

[36] W. Wang, Y. Wang and Z. Cao, "Dynamic Soft Real-Time Scheduling with Preemption Threshold for Streaming Media," *International Journal of Digital Multimedia Broadcasting,* vol. 2019, no. 1, p. DOI: 10.1155/2019/5284968, 2019.

[37] A. Silberschatz, B. G. Peter and G. Greg, Operating System Concepts, 8th ed., Wiley, July 29, 2008.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy) The authors equally contributed to the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself: This research is funded by the Deanship of Research and Graduate Studies in Zarqa University /Jordan.

Conflicts of Interest The authors have no conflicts of interest to declare that are relevant to the content of this article.