Comparison Analysis of Round Robin Algorithm with Highest Response Ratio Next Algorithm for Job Scheduling Problems

Benny Richardson, Wirawan Istiono

***Abstract*—Job Scheduling is the process of allocating operating system resources to do several different jobs. The problem that arises is how to manage the work of the system to complete a lot of existing work in a timely and optimal time. To solve the problem, several heuristics and metaheuristics are used. The goal is to minimize the total time that all work has been executed. In this paper, the researcher will compare the Round Robin algorithm and the Highest Response Ratio Next algorithm to find which algorithm is the most optimal for completing all work on time. In this research, a comparison test was conducted with three cases, and judging from the 3 cases that have been compared between the Round Robin algorithm and High Response Ratio Next, it was found that the High Response Ratio Next algorithm is more optimal than the process carried out using the Round Robin algorithm. Due to the waiting time for each process that is run by the CPU, the High Response Ratio Next algorithm is less than the waiting time in the Round Robin algorithm.**

***Keywords*— Job Scheduling Problems, Round Robin, Highest Response Ration Next, Quantum Time, Burst Time**

I. INTRODUCTION

In this modern era, everything becomes easy to do with the help of technology. People no longer have to bother using conventional methods to do their work because by using technology, their work can be completed in a short time. Although people do not understand using technology, it doesn't hamper their work because this increasingly sophisticated technology can help people with an ability designed by humans themselves so that the technology can have the ability to understand, think and do their own work [1].

So with this convenience, more needs are needed and cause more work to be done. Moreover, the work must be completed on time and not cause delay. This will certainly be difficult if not regulated properly. Therefore, we need a scheduling system on the job to choose which work must be done in advance so that all existing work can be completed on time [2]. Job scheduling has an important role in planning and managing the manufacturing process. There are various kinds of algorithms used for scheduling [3], [4]. In this study, using two algorithms, namely the Round Robin algorithm and Highest Response Ratio Next algorithm, to compare which algorithm can complete a collection of work in a timely manner with the fastest optimal time.

II. JOB SHOP SCHEDULING PROBLEM

Job Scheduling is the process of allocating system resources to several different jobs performed by the operating system (OS). The system will handle the priority work queue by waiting for the CPU time to do the job and must determine the work to be taken from seeing which queue and the amount of time allocated for the job [5], [6]. This work is carried out to ensure that all work is carried out fairly and on time. The schedule for each job is allocated one or several time intervals for one or more machines.

Recently, researchers have focused on investigating the problem of scheduling machines in manufacturing and service environments where work represents activity and machines represent resources. Each machine can process one job at a time. The problem in job scheduling is a complex computational process and it is difficult to find an optimal solution in time efficient because the increase in the search space grows exponentially as the number of job queues increases and can cause many delays on some job [3], [5].

In the case of work scheduling, every work is assumed to be in condition in this criteria:

- All jobs not depend on each other.
- Each job can only be processed by one machine.
- No Jobs are processed at the same time on the same machine.
- Jobs that have a high priority value must take precedence.
- Jobs that have less time take precedence.
- The total time the work has been processed must be in optimal time.

The Round Robin algorithm is a simple scheduling algorithm and is most used in timeshared systems. Each process is given a time interval called quantum time, as long as the process is in quantum time, allowed to run [7], [8]. This algorithm selects the process from the front row and runs the process during the given quantum time. The new process will be added to the back ready queue. One of two things will happen, this process may have more burst time than quantum time. In this case, the process will run during the given quantum time and the remainder of the process will be put back into the ready queue [9][10]. If the process ends for less than the given amount of time, choose another

process from the ready queue and run the process at most one time. In Round Robin, there is no process allocated by the CPU for more than the given quantum time [11]. The purpose of this algorithm is to use resource allocation and also produce a minimum response time compared to the Shortest Job First and First Come First Serve algorithm. Scheduling with Round Robin assumes all important processes [12].

The advantage of the Round Robin algorithm is that each process is carried out by the CPU for a fixed time so that the priority of each work is the same and starvation does not occur because of its cyclic nature [13]. The drawback is the result depends on quantum time and cannot be given priority on every job.

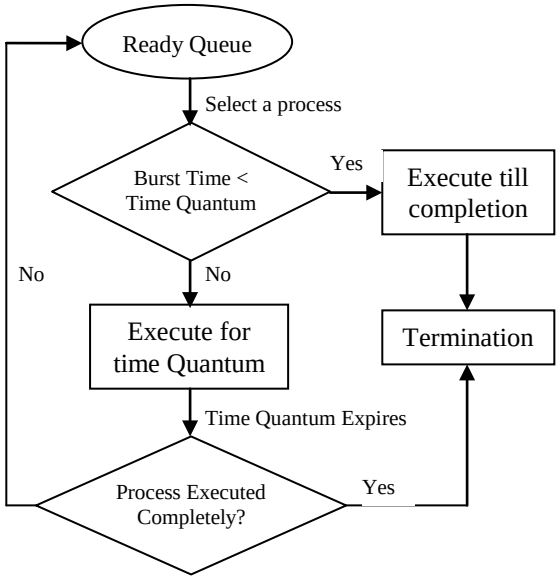


Fig. 1. Round Robin Scheduling Flowchart

Steps in the Round Robin algorithm [14]:

1. Begin the process.
2. Make a ready queue of jobs that request to the CPU.
3. Do that job.
4. Choose the first process to enter the ready queue and allocate the CPU to do the work at the time interval during the given quantum time.
5. If the remaining CPU burst time from work that is running is less or equal to the given quantum time, reallocate the CPU to the work in progress for the remaining CPU burst time. After the job is finished, the job is no longer put in the ready queue.
6. If more than quantum time is given, delete the current job from the queue and reinsert the job to the end of the ready queue.
7. Choose the job with the next shortest burst time from the queue and CPU allocation to do the work during the given quantum time. Then back to step 5.
8. Doing repetition to check the ready queue is empty or not.
9. Calculate average waiting time, average turnaround time, and number of contexts.
10. End.

III. HIGHEST RESPONSE RATIO NEXT ALGORITHM

The Highest Response Ratio Next algorithm (HRRN) is an algorithm whose priority for each job depends on the estimated running time, and based on the amount of time spent waiting [15]. Jobs get higher priority the longer the work is postponed (starvation process). Even long jobs are delayed by CPUs competing with jobs that are estimated to have a short period of time [16], [17]. The CPU is assigned to work on the process that has the highest response ratio using the following formula [18].

Priority = waiting time + estimated runtime

estimated runtime

OR

$$Ratio = \frac{time\ spent\ waiting + expected\ service\ time}{expected\ service\ time}$$

Fig. 2. Highest Response Ratio Next Formula

The advantage of this algorithm is that it increases SPF scheduling, still Non-pre-emptive, can consider how long the process has been waiting for, preventing unlimited delays. The disadvantage of this algorithm is that it does not support the system's external priority, the process is scheduled to use the system's internal priorities [19]. This is an Algorithmic procedure, first to begin process, the first step is create a queue of jobs requesting to the CPU and then selects the first process that enters the queue and allocates the CPU to do the work at time intervals during the given time quantum. If the remaining CPU burst time of the running job is less or equal to the given quantum time, re-allocate the CPU to the running job for the remaining CPU burst time. Once the job has finished executing, the job is no longer queued. If more than the given quantum time, remove the running job from the queue and reinsert the job to the end of the queue. And then select the job with the next shortest burst time from the queue and allocate CPU to do the job during the given quantum time. After that repeat to check whether the queue is empty or not and calculate the average waiting time, average turnaround time, and number of contexts [16].

IV. RESEARCH METHOD

A. Figures and Tables

The research will be conducted by comparing the Round Robin algorithm and Highest Response Ratio Next of the three cases given to find which algorithm is the most optimal in carrying out all processes that enter the CPU. The study was conducted using C and C # programming languages. In Fig 3 shown the pseudocode procedure of Round Robin. And in the Fig 4 shown the pseudocode procedure of Highest Response Ratio Next Algorithm.

```
Initial ready_queue process, Arival_Time, Burst_Time,
Quantum_Time

while(ready_queue != null)
    Execute task in Round Robin manner
```

```
    If(Burst_time of task > Quantum_time)
        If(other task on ready_queue during interval CPU time)
            Put recent task after other task in ready_queue
        else
            put recent task on ready_queue
    else
        go to other task in ready_queue
end while
computer Average waiting time
compute Avaerage compilation time
```

Fig. 3. Pseudocode Round Robin Algorithm

```
Initial ready_queue process, Arival_Time, Burst_Time, Ratio,
time_spent_waiting, Highest_Ration, Index_Highest_Ratio

While (ready_queue != null)
    Execute task in Highest Response Ration Next manner
    Check other task in ready queue
        Do i to amount of task in ready_queue
            Time_spent_waiting = Max interval time now –
Arival_time of task[i]
            Ratio task[i] = (time_spent_waiting + burst_time of
task[i]) / burst_time of task[i]
        if(ratio task[i] > Highest_Ratio)
            Highest_ratio = Ratio task[i]
            Index_Highest_Ratio = i
        End Do
    Go to task[Index_Highest_Ratio]
End While
Compute Average waiting time
Compute Average Turn Around Time
```

Fig. 4. Pseudocode Highest Response Ratio Next Algorithm

V. Result And Analysis

Based on pseudocode procedure Round Robin algorithm and Highest Response Ratio Next Algorithm that shown in Fig3 and Fig4, do the test with three cases. The first case is burst time is given randomly with a quantum time of 3s. In Table 1 is shown sample cases with random burst time.

Table 1. Random Burst time

Process	AT	BT
P0	0	8
P1	5	3
P2	7	7
P3	4	5
P4	1	4
P5	9	2

AT: arrival time of the work in ready queue
BT: the amount of time the CPU will execute (Burst time)
QT: quantum time

In Round Robin algorithm put the work in queue; P0, P1, P2, P3, P4, and P5. Each process is given a quantum time (TQ) of 3s periodically. And after that create Gantt chart like shown in Fig 5.

P0	P0	P3	P4	P1	P0	P4	P2	P3	P5	P2	P2

0 3 6 9 12 15 17 18 21 23 25 28 29

Fig 5. Gantt chart Cases 1

And after create Gantt chart for first cases, and implement pseudocode procedure Round Robin algorithm, got the result that, the average waiting time is 14, and the average compilation time is 18.83333 as shown in Fig 6.

```
name compile_time waiting_time
 p0      19              11
 p1      12              9
 p2      24              17
 p3      21              16
 p4      19              15
 p5      18              16

Average waiting time is 14
Average compilation time is 18.83333
Sequence is like that ->p0->p0->p3->p4->p1->p0->p4->p2->p3->p5->p2->p2
```

Fig. 6. Results of compilation of Round Robin case 1

In Highest Response Ratio Next algorithm put the job in the ready queue: P0, P1, P2, P3, P4, and P5. Because it is Non-pre-emptive, the CPU runs the P0 process completely. It takes 8ms to execute the p1 process. Then between the processes P1, P2, P3, P4 with the highest response ratio will be chosen to be executed. The calculation is shown in Fig 7 below.

Ratio for P1 = ((8-5) + 3) / 3 = 2

Ratio for P2 = ((8-7) + 7) / 7 = 1.143

Ratio for P3 = ((8-4) + 5) / 5 = 1.8

Ratio for P4 = ((8-1) + 4) / 4 = 2.75

P0	P4

0 8 12

Fig 7. First step calculation Highest Response Ration Next algorithm in cases1

The next process that will be done is the process with the highest ratio of processes P1, P2, P3, and P5, that shown in Fig 8 below.

Ratio for P1 = ((12-5) + 3) / 3 = 3.333

Ratio for P2 = ((12-7) + 7) / 7 = 1.714

Ratio for P3 = ((12-4) + 5) / 5 = 2.6

Ratio for P5 = ((12-9) + 2) / 2 = 2.5

P0	P4	P1

0 8 12 15

Fig 8. Second step calculation Highest Response Ration Next algorithm in cases1

Doing the search of ratio again to do the next job. Here are the final results of the Gant Chart obtained, that shown in Fig 9.

P0	P4	P1	P5	P3	P2

0 8 12 15 17 22 29

Fig 9. Final step calculation Highest Response Ration Next algorithm in cases1

And after create Gantt chart for first cases, and implement pseudocode procedure Highest Response Ration Next algorithm, got the result that, the average waiting time is 8, and the average compilation time is 12.83333 as shown in Fig 10.

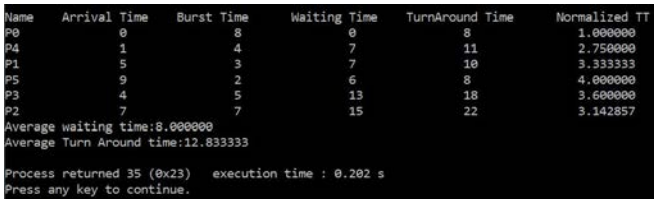


Fig. 10. Results of compilation of Highest Response Ratio Next case 1

In the second case, burst time increases ascending with a quantum time of 3s. In Table 2 is shown the value for second cases.

Table 2. Burst time increases ascending

Process	AT	BT
P0	0	2
P1	5	3
P2	7	4
P3	4	5
P4	1	7
P5	9	8

In Round Robin algorithm put the work in queue; P0, P1, P2, P3, P4, and P5. Each process is given a quantum time (TQ) of 3s periodically. And after that create Gantt chart like shown in Fig 11.

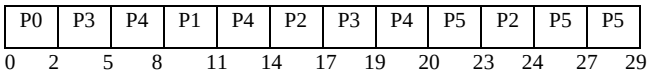


Fig 11. Gantt chart Cases 2

And after create Gantt chart for second cases, and implement pseudocode procedure Round Robin algorithm, got the result that, the average waiting time is 14, and the average compilation time is 18.83333 as shown in Fig 12.

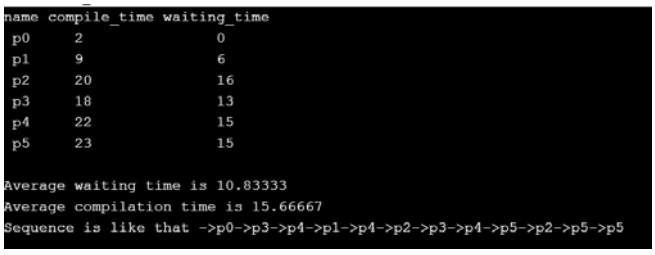
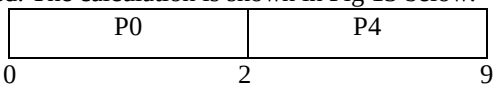


Fig. 12. Results of compilation of Round Robin case 2

In Highest Response Ratio Next algorithm, same process like HRRN step before, first, put the job in the ready queue: P0, P1, P2, P3, P4, and P5. Because it is Non-pre-emptive, the CPU runs the P0 process completely. It takes 8ms to execute the p1 process. Then between the processes P1, P2, P3, P4 with the highest response ratio will be chosen to be executed. The calculation is shown in Fig 13 below.

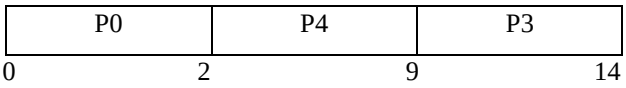


Ratio for P1 = ((9-5) + 3) / 3 = 2.333

Ratio for P2 = ((9-7) + 4) / 4 = 1.5

Ratio for P3 = ((9-4) + 5) / 5 = 2

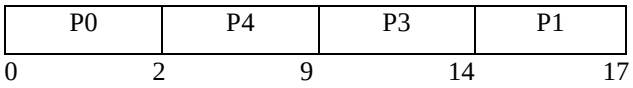
Ratio for P5 = ((9-9) + 8) / 8 = 1



Ratio for P1 = ((14-5) + 3) / 3 = 4

Ratio for P2 = ((14-7) + 4) / 4 = 2.75

Ratio for P5 = ((14-9) + 8) / 8 = 1.625



Here are the final results of the Gant Chart obtained.

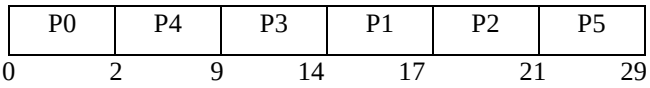


Fig. 13. Calculation step of Highest Response Ratio Next algorithm in case 2

And after create Gantt chart with Highest Response Ratio Next algorithm for second cases, and implement pseudocode procedure, the result got, the average waiting time is 6.166667, and the average Turn Around time is 11 as shown in Fig 14.

Fig. 14. Results of compilation of Highest Response Ratio Next case 2

In the third case, decreases descending with a quantum time of 3s. In Table 3 is shown sample value for cases 3.

Table 3. Burst time decreases descending

Process	AT	BT
P0	0	8
P1	5	7
P2	7	5
P3	4	4

P4	1	3
P5	9	2

In Round Robin algorithm put the work in queue; P0, P1, P2, P3, P4, and P5. Each process is given a quantum time (TQ) of 3s periodically. And after that create Gantt chart like shown in Fig 15.

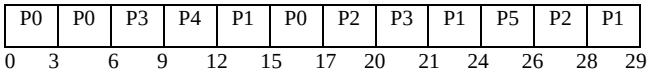


Fig 15. Gantt chart Cases 3

And after create Gantt chart for second cases, and implement pseudocode procedure Round Robin algorithm, got the result that, the average waiting time is 15, and the average compilation time is 19.83333 as shown in Fig 16.

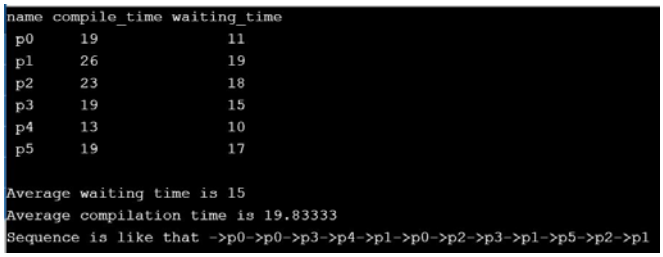
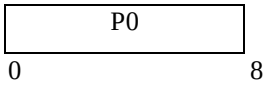


Fig. 16. Results of compilation of Round Robin case 3

In Highest Response Ratio Next algorithm, the same process is carried out as the previous HRRN step. And in Fig 17 can be seen the calculation step of Highest Response Ratio Next algorithm for case 3.

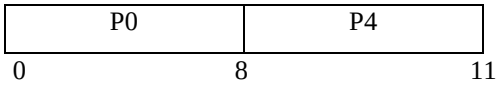


Ratio for P1 = ((8-5) + 7) / 7 = 1.429

Ratio for P2 = ((8-7) + 5) / 5 = 1.2

Ratio for P3 = ((8-4) + 4) / 4 = 2

Ratio for P4 = ((8-1) + 3) / 3 = 3.3333

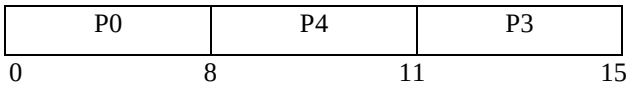


Ratio for P1 = ((11-5) + 7) / 7 = 1.857

Ratio for P2 = ((11-7) + 5) / 5 = 1.8

Ratio for P3 = ((11-4) + 4) / 4 = 2.75

Ratio for P5 = ((11-9) + 2) / 3 = 1.3333



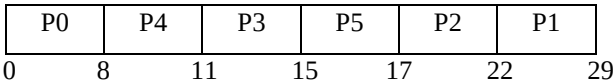


Fig. 17. Calculation step of Highest Response Ratio Next algorithm in case 3

And after create Gantt chart with Highest Response Ratio Next algorithm for second cases, and implement pseudocode procedure, the result got, the average waiting time is 7.833333, and the average Turn Around time is 12.666667 as shown in Fig 18.

Fig. 18. Results of compilation of Highest Response Ratio Next case 3

There can be seen from the three cases that have been done, the process that is run using the High Response Ratio Next algorithm is more optimal than the process that is run using the Round Robin algorithm. Evidenced by the waiting time for each process run by the CPU, the High Response Ratio Next algorithm is less than the waiting time on the Round Robin algorithm.

VI. CONCLUSION

Based on the results obtained in this study, it can be seen that the average waiting time of each job done by the CPU using the Round Robin algorithm is much longer than the High Response Ratio Next algorithm. This is because in the Round Robin algorithm, the CPU can be interrupted by other processes that enter the queue. Whereas the High Response Ratio Next algorithm does not use quantum time, the process that enters the queue will be done until the process is complete. After completing it, just doing another process entered, but in this algorithm see the priority of each process by calculating the ratio of each process. The Round Robin algorithm does not work on a process based on that process. High or little quantum time and burst time each process has no effect on the speed of the Round Robin algorithm in completing all processes when compared with the speed of the Highest Response Ratio Next in completing all processes that enter the CPU queue.

VII. FUTURE WORK

To improve accuracy average waiting time and turn around, can adding some input parameter, such as job sequencing with deadline, or can be try comparison other algorithm job scheduling to get best job scheduling algorithm.

ACKNOWLEDGMENT

Thank you to the Universitas Multimedia Nusantara, Indonesia which has become a place for researchers to develop this journal research. Hopefully, this research can make a major contribution to the advancement of technology in Indonesia.

REFERENCES

[1] S. Kavitha, P. Venkumar, N. Rajini, and P. Pitchipoo, "An Efficient Social Spider Optimization for Flexible Job Shop Scheduling Problem," *Journal of Advanced Manufacturing Systems*, vol. 17, no. 2, pp. 181–196, 2018.

[2] D. D. Sinaga and S. Hansun, "Indonesian text document similarity detection system using rabin-karp and confix-stripping algorithms," *International Journal of Innovative Computing, Information and Control*, vol. 14, no. 5, pp. 1893–1903, 2018.

[3] Y. Habibi, G. Swalaganata, and A. D. Yustita, "Penyelesaian Multi-Objective Flexible Job Shop Scheduling Problem Menggunakan Hybrid Algoritma Imun," *Jurnal Teknosains*, vol. 6, no. 2, p. 79, 2017.

[4] S. A. Shukor, I. M. Shaheed, and S. Abdullah, "Population initialisation methods for fuzzy job-shop scheduling problems: Issues and future trends," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 4–2, pp. 1820–1828, 2018.

[5] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu, "Review of job shop scheduling research and its new perspectives under Industry 4.0," *Journal of Intelligent Manufacturing*, vol. 30, no. 4, pp. 1809–1830, 2019.

[6] J. Alzahrani, "Job-Shop Scheduling Optimization With Stochastic Processing Times," *International Journal of Engineering Technologies and Management Research*, vol. 6, no. 1, pp. 73–83, 2020.

[7] T. Hidayat, Y. Azzery, and R. Mahardiko, "Load Balancing Network by using Round Robin Algorithm: A Systematic Literature Review," *Jurnal Online Informatika*, vol. 4, no. 2, p. 85, 2020.

[8] Awang and F. Badri, "Dual Load balancing menggunakan Algoritma Round-robin pada Cloud Computing," *Applied Technology and Computing Science Journal*, vol. 2, no. 2, pp. 70–78, 2020.

[9] C. KOMALASARI and W. ISTIONO, "A Comparative Study of Cocktail Sort and Insertion Sort," *Journal of Applied Computer Science & Mathematics*, vol. 15, no. 1, pp. 21–25, 2021.

[10] S. Ul Sabha, "A Novel and Efficient Round Robin Algorithm with Intelligent Time Slice and Shortest Remaining Time First," *Materials Today: Proceedings*, vol. 5, no. 5, pp. 12009–12015, 2018.

[11] G. P. Arya, K. Nilay, and D. Prasad, "An improved round robin CPU scheduling algorithm based on priority of process," *International Journal of Engineering and Technology(UAE)*, vol. 7, no. 4, pp. 238–241, 2018.

[12] A. Joshi, "A Modified Version of Round Robin Algorithm ' Modulo Based Round Robin Algorithm ,'" *International Journal of Advanced Science and Technology*, vol. 29, no. 11, pp. 576–578, 2020.

[13] S. Zouaoui, L. Boussaid, and A. Mtibaa, "Improved time quantum length estimation for round robin scheduling algorithm using neural network," *Indonesian Journal of Electrical Engineering and Informatics*, vol. 7, no. 2, pp. 190–194, 2019.

[14] A. Fiad, Z. M. Maaza, and H. Bendoukha, "Improved version of round robin scheduling algorithm based on analytic model," *International Journal of Networked and Distributed Computing*, vol. 8, no. 4, pp. 195–202, 2020.

[15] R. K. Sanodiya, S. Sharma, V. Sharma, R. Gandhi, P. Vishwavidyalaya, and B. M. P -India, "A Highest Response Ratio Next(HRRN) Algorithm Based Load Balancing Policy For Cloud Computing," *International Journal of Computer Science Trends and Technology*, vol. 3, no. 1, pp. 72–76, 2013.

[16] S. Raheja, R. Dhadich, and S. Rajpal, "Vague Oriented Highest Response Ratio Next (VHRRN) scheduling algorithm," *IEEE International Conference on Fuzzy Systems*, 2013.

[17] R. Latip and Z. Idris, "Highest Response Ratio Next (HRRN) vs First Come First Served (FCFS) scheduling algorithm in grid environment," *Communications in Computer and Information Science*, vol. 180 CCIS, no. PART 2, pp. 688–693, 2011.

[18] P. S. Varma, "Design of Modified HRRN Scheduling Algorithm for priority systems Using Hybrid Priority scheme," *Journal of Telematics and Informatics*, vol. 1, no. 1, 2013.

[19] G. A. Shidali, S. B. Junaidu, and S. E. Abdullahi, "Weighted Pre-Emptive Modified Highest Response Ratio Next," *Computer Science and Engineering*, vol. 5, no. 3, pp. 59–65, 2015.

Benny Richardson, was born in Jambi, Indonesia. He received the bachelor's degree in Informatics from Universitas Multimedia Nusantara, in 2020. His research interests include a user and learning based web and mobile programming

He now works as Software Engineer in Mobilepulsa, Tangerang Indonesia from June 2020. And before that, he has worked as Laboratory Assistant in Universitas Multimedia Nusantara, Tangerang Indonesia.

His last publication entitled is "Penerapan Konsep Non-Deterministic Finite Automata (NFA) pada Aplikasi Simulasi Mesin Kopi Vending" in 2019. Email: benny.richardson@student.umn.ac.id

Wirawan Istiono, was born in Jakarta Indonesia, he received his Masters of Computer Science degree in Budi Luhur University, Indonesia, in 2018, focusing in the Software Engineering field.

He is currently a lecturer and researcher in Universitas Multimedia Nusantara and also serving as the head coordinator of the Game Development Laboratory. His research interests include requirements engineering in software application development, computer engineering, and human computer interaction.

His last publication entitled "Virtual Reality Game for Introducing Pencak Silat" in conference Second International on Informatics and Computing (ICIC) in 2017 at IEEE. And last publication entitled "Virtual Reality Game for Introducing Pencak Silat" in International Journal of Interactive Mobile Technologies (IJIM) in 2020. Email: wirawan.istiono@umn.ac.id