CHAPTER ONE

BACKGROUND

1.1 Introduction

Operating system manages and coordinates the functions performed by computer hardware as well as software. The main function of the operating system as a resource manager and interface between hardware and user.

The operating system performs basic tasks, such as controlling and allocating memory, prioritizing the processing of instructions, controlling input and output devices, facilitating networking, and managing files. Modern operating systems have been becoming more and more complex. They have evolved from a single task to multitasking environment in which processes run in a concurrent manner [1]. CPU scheduling is an essential operating system task; therefore its scheduling is central to operating system design. When there is more than one process in the ready queue waiting its turn for CPU, the operating system must decide through the scheduler the order of the executions of ready processes. Allocating CPU to process requires careful attention to assure fairness and avoid process starvation for CPU. Scheduling decision try to minimize the followings parameters:

- Turnaround time
- Response time
- Average waiting time for processes
- And number of context switching

1.2 Long TermVs Short Term Scheduling

The aim of processor scheduling is to assign process to be executed by the processor over time, in a way that meets the system objectives, such as response time, throughput and processor efficiency [2]. The long-term scheduler also known as admission scheduler determines which jobs/processes is to be submitted to the ready queue in the Main Memory that is, when an attempt is made to execute a program, its admission to the set of currently executing processes is either authorized or postponed by the long-term

scheduler. This type of scheduler determines that which jobs/processes are to be run on a system, the degree of concurrency to be supported at a time and also plays a vital role for concurrently running high amount of processes and how to categorize between I/O bound and CPU bound processes. The long term scheduler is responsible for controlling the degree of multiprogramming. Long-term scheduling is also important in large-scale systems such as batch processing systems, Computer Cluster, "Supercomputer" and render farms. In these cases, special purpose job scheduler software is typically used to assist these functions, in addition to any underlying admission scheduling support in the operating system.

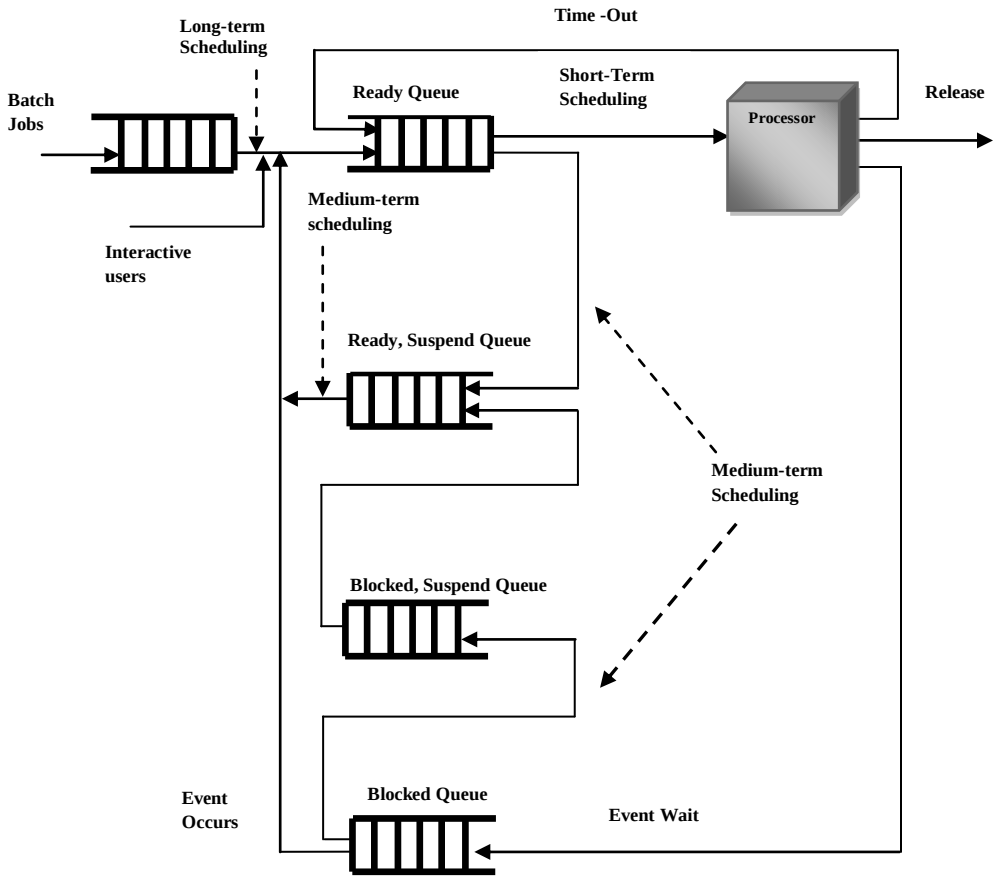


Fig. 1.1: Queuing diagram for scheduling [2]

Medium-term scheduling is the part of swapping function. Scheduler temporarily removes processes from main memory and places them on secondary memory or vice versa. This is commonly referred to as "swapping out" or "swapping in" (also incorrectly

as "paging out" or "paging in"). The medium-term scheduler decide to swap out a process which has not been active for some time, or a process which has a low priority, or a process which is page faulting frequently, or a process which is taking up a large amount of memory in order to free up main memory for other processes, swapping the process back in later when more memory is available, or when the process has been unblocked and is no longer waiting for a resource.

The medium-term scheduler may actually perform the role of the long-term scheduler, by treating binaries as "swapped out processes" upon their execution. In this way, when a segment of the binary is required it can be swapped in on demand, or "lazy loaded".

The short-term scheduler also known as the CPU scheduler or dispatcher executes most frequently and makes the fine-grained decision of which process to execute next. This scheduler decides which of the ready, in-memory processes are to be executed (allocated a CPU) after a clock interrupt, an I/O interrupt, an operating system call or another form of signal. Thus the short-term scheduler makes scheduling decisions much more frequently than the long-term or mid-term schedulers - a scheduling decision will at a minimum have to be made after every time slice and these are very short. This scheduler can be preemptive, implying that it is capable of forcibly removing processes from a CPU when it decides to allocate that CPU to another process, or non-preemptive (also known as "voluntary" or "co-operative"), in which case the scheduler is unable to "force" processes off the CPU.A preemptive scheduler relies upon a programmable interval timer which invokes an interrupt handler that runs in kernel mode and implements the scheduling function.

1.3 Batch processing operating systems

Executing a series of non- interactive jobs all at one time. Batch processing is particularly useful for operations that require the computer or a peripheral device for an extended period of time. Once a batch job begins, it continues until it is done or until an error occurs. Note that batch processing implies that there is no interaction with the user while the program is being executed.

An example of batch processing is the way that credit card companies process billing. The customer does not receive a bill for each separate credit card purchase but one monthly bill for all of that month's purchases. The bill is created through batch processing, where all of the data are collected and held until the bill is processed as a batch at the end of the billing cycle.

1.4 Multitasking interactive operating system

In multitasking operating system multiple processes are executed during the same period of time, processes run concurrently instead of sequentially. The processes share the resources like CPU and memory.

1.5 Real time operating system

Real-time operating systems are systems that respond to the input immediately or with a specified deadline. This category includes operating systems designed substantially for the purpose of controlling and monitoring external activities with timing constraints.

1.6 Program and processes

Process is the program in execution. A program is combination of instructions and data together to form a task when executed. Each process has its own address space, which typically consists of program parts and data parts. The program part contains data required for the process, it also contains the state of the process such as contents of the CPU registers which change dynamically with the execution of the instructions.

1.6.1 Process States

When the process is loaded in the memory, it becomes ready for execution. when the scheduler selects the process for execution, the process enters the running state. In this state the process can either be preempted which is the case when it exceeds the time quantum allocated or blocked while waiting for I/O data. When process is preempted then the operating system puts the process on the end of the ready queue of processes, but it remains ready to execute , if the process is blocked while waiting for IO operation, it is then taken from ready queue and put on the I/O queue. When I/O channel completes the I/O operation for blocked process, the process reenters the ready state, where it wait for CPU. Thus at any time each process may be in one of the following states. The ready state, Where the process is ready to run, and waiting for CPU. From this state process

can enter the running state. In the process is using the CPU, and process can either be preempted and put in the ready state , or may go blocked state for I/O operation or may terminate with or without error. In the blocked state the process is waiting for I/O operation. When channel completes its I/O operation, then the process becomes ready and the operating system puts it in the back of the ready queue.

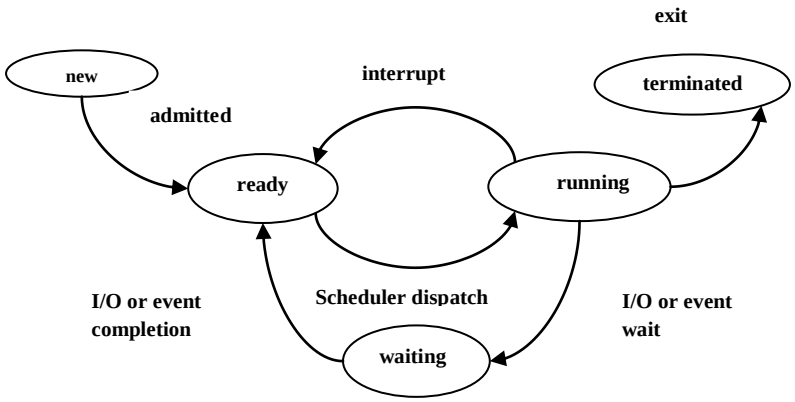


Fig.1.2 Diagram of process states

1.7 Scheduling

Scheduling is the fundamental function of the operating system. CPU Scheduling deals with the problem of deciding which process is selected from the ready queue for the CPU. When the CPU becomes idle, the operating system must select process from ready queue in the ready queue. The part of the operating system which makes the choice for the process in the ready queue runs next is called scheduler and the algorithm it uses is called scheduling algorithm.

There are several primitive scheduling algorithms such as first-come first serve, shortest job first , Priority based, Round-Robin, Multilevel Queue, Multilevel feedback queue etc. However there is no one is best scheduling algorithm, each has its own characteristics.

Round Robin is basically first come first serve algorithm and known as simple and fairest among the processes. Several research has been done in Round Robin scheduling algorithm. It is used in variety of systems like packet switching ,real time systems, multiprogramming , time sharing systems environment.

1.8 Quanta Size Vs Context Switch

A time quantum is generally a small amount of time, range from 10 to 100 milliseconds [3]. A quantum though is normally not a fixed value, and may vary with process priority, with time and other factors. How the quantum value is determined is a policy decision of the operating system and not of the scheduler.

The performance of RR scheduling depends on the size of time quantum. If the quantum value is too large, it behaves as FCFS scheduling algorithm. If the quantum value is too low, context switches are too frequent causing additional burden of workload because in this time period no purposeful computation is carried out for any process. The usual range for the quantum value between 20 to 50 times of the context switch time. If the both the quantum size and the context switch overhead are close to zero. The RR is called processor sharing because, in theory, it appears to the processes that each has its own, though slower, CPU. All ready processes have equal share of CPU time and their speed is inversely proportional to the number of processes in the ready queue.

1.9 Motivation

CPU scheduling is a very important task of the operating system. Different processes need to allocate CPU for different amount of time. In interactive system this allocation of CPU among the different processes needs to be primitive as no process should allocate CPU forever. Studies and researches have shown that RR is one of the simplest, fairest and most widely used algorithm. In RR each process is assigned a time interval, this time interval is called quantum of the process during which the process is allowed to use the CPU. The size of this quantum is the main factor for different algorithms. Different algorithms has been designed and implemented to estimate the best choice of time quanta, some of these claimed that their approach is best, so this dissertation work is mainly focused on the comparison and analysis of some these approaches.

1.10 Problem statement

The size of the time quantum is the crucial issue in Round Robin scheduling. This dissertation work focuses on comparing different algorithms by varying time quantum and studying the effects of these time quanta on the different parameters. It also aims to estimate the best time quanta.

1.11 Objectives

- To study about the variants of RR scheduling algorithms.
- To compare different dynamic RR algorithms in terms of scheduling parameters.
- To Analyze/Suggest a better Size of time quanta for RR.

1.12 Structure of thesis

This study is organized into six chapters. The first chapter deals with background which includes introduction , motivation, problem statement, objectives and structure of thesis. The second chapter deals with literature review which includes short description of the algorithms under study. The third chapter describes research methodology, it includes experimental framework, assumptions, some input and out parameters and tracing algorithms with different parameters, tools, programming language, IDE, data structures , flow charts and algorithms. The fourth chapter includes results, analysis and comparison. The fifth chapter is conclusion part. At last the chapter six presents recommendations and limitation of the study.

CHAPTER TWO

LITERATURE REVIEW

Different research has been done in the field of CPU scheduling and many of these researches are focused on the optimization of the time quanta. RR is one of the most widely explored algorithm.

Noon et al. [4] have proposed a new algorithm based on dynamic time quantum. In this the time quantum are adjusted by the operating system according to the burst time of the set of waiting processes in the ready queue. When the ready queue is empty, the TQ is equal to the BT of the first process loaded in the ready queue. When the ready queue is not empty then the TQ is calculated by using mean average of the sum of burst time of processes in the ready queue. This approach claims to solve the fixed time quantum problem and increase the performance of RR.

Nayak, et al. [5] have introduced the idea of making repeated adjustment of time quantum according to the burst time of the current running processes.

Here, the processes are arranged in ascending order according to their burst time present in the ready queue. The optimal time quantum is calculated as follows:

$$\text{Optimal Time Quantum} = \frac{\mathbf{HighestBT + Median}}{\mathbf{2}}$$

Where Highest BT is the largest burst time value in the current ready queue and median is calculated as follows:

$$\text{Median} = \begin{cases} \mathbf{Y(n+1)/2\ when\ n\ is\ odd} \\ \frac{\mathbf{1}}{\mathbf{2}}\mathbf{((Yn/2) + (Y1 + n/2))\ when\ n\ is\ even} \end{cases}$$

Where, Y is the number located in the middle of a group of numbers arranged in ascending order and n is the number of processes.

The optimal time quantum is assigned to each process and is recalculated taking the remaining burst time in account after each cycle. This procedure goes on until the ready

queue is empty. They have shown that their algorithm solves the fixed time quantum problem and supports building of self-adaption operating system.

Panda et al. [7] have considered different TQ for a group of processes and proposed a group based time quantum algorithm. In their work, the processes are sorted in RQ. The quartile measure is used to form a group among the processes. The Q1 is the 25% of the data set. The Q2 (or median) is the 50% of the data set. Finally, the Q3 is the 75% of the data set. Based on the CPU BT, the processes are formed four groups. Each group has different TQ. Different TQ is used to reduce CS. Min-Max dispersion or spread measure was taken to calculate the TQ. It may suffer from CS, if the difference between MaxBT and MinBT is very less. So, in this algorithm, α is used as a threshold to reduce CS. Finally, the TQ is assigned to each group. The process is continued until RQ is empty. After execution of all processes, ATAT, AWT and CS are calculated.

AshimGhishing [6] analyzes the changes and impacts on performance of batch multiprogrammed system (MOS) with varying time quantum round robin scheduling over the fixed time quantum RR and show the better performance of dynamic time quantum over fixed time quantum.

Rami J. Mathernesh [8] Presents a solution to the time quantum according to the burst time of the existed set of processes in the ready queue. The optimal time quantum is calculated by the median of the set of processes in the ready queue, if the median is less than 25 then its value is set to 25 so the overhead of the context switch is avoided. P.Banerjee, et al. [9] developed a scheduling algorithm called average max round robin algorithm. The time quantum is mean of the sum of the average and maximum burst time. S.M. Mostafa et al. [10] presents time quantum of Round Robin CPU scheduling algorithm in general computing system using integer programming. This paper proposed the integer programming to solve equation that decides a value that is neither too large nor a too small such that every process has reasonable response time and the through put of the system is not decreased due to unnecessary context switches. The time quantum is changed in each round over the cyclic queue. P.S. Varma [11] give the comprehensive study and analysis of Round Robin algorithm and SRBRR algorithm. He proposed an improve version SRBRR (Shortest Remaining Burst Time RR) by assigning the processor to process with the help of median and highest burst time. This experimental

analysis shows that ISRBRR performs better than RR algorithm, as SRBRR in term of reducing number of context switch, average waiting time, average turnaround time.

S. Saeidi, et al. [12] developed a nonlinear mathematical model for optimizing the time quantum value in RR scheduling algorithm. In this research mathematical concept is used to obtained the optimum value for time quantum parameter in order to minimize the average waiting time of the processes. Behera et.al. [13] also proposed a method called Multi Dynamic Time Quantum Round Robin (MDTQRR) in which TQ is calculated dynamically. Albielmona [1] presented on extensive review of different scheduling algorithm. Nieh et al. [14] proposed a new scheduling algorithm known as virtual time round robin (VTRR) with complexity of o(1) by combining fair queuing algorithms with the RR scheduling algorithm. Zahedi et al. [15] used the concept of fuzzy rules to optimize the RR algorithm and introduced fuzzy rule based round robin CPU scheduling (FRRCS). They showed that the proposed algorithm, reduce the average waiting time. C. Yaashuwanth, and R. Ramesh [16] purposed algorithm in which the time quantum is calculated intelligently and individually for each process. The throughput of the system is increased due to the decreasing the total number of context switches. Siregar [17] has purposed a method of determining optimum time quantum value by the use of genetic algorithm and tries to minimize the average waiting time of the process.

CHAPTER –THREE

RESEARCH METHODOLOGY

3.1 Experimental Framework

To evaluate the performance of different algorithm, I have taken a set of even number of processes in different cases. Here for simplicity, 10 processes are taken. The algorithm works effectively even if it used with a very large number of processes.

3.2 Assumption

- All the processes are assumed to be independent
- The time slice is assumed to be not more than maximum burst time.
- All the attributes, like number of processes, burst time and time slice are known before submitting the processor.
- All processes are CPU bound not I/O bound.

Inputs:

- Burst time (Bt)
- Arrival time (At)
- Optimal time quantum (Otq)
- Number of processes (pn)
- Time for one cycle (Q) for DRR

Output:

- Average waiting time (Awt)
- Average turnaround time (Att)
- Number of context switch.

3.3 Studied Algorithms

DRR Scheduling Algorithm

For this algorithm firstly the optimum quantum value is calculated by running static round robin algorithm by varying the quantum from 1 to maximum burst time of process in ready queue then the quantum is observed for the minimum number of context switch and minimum average turnaround time. Based on this value the quantum for each cycle is dynamically calculated according to the jobs remaining in the ready queue.

Mean Based Algorithm

The process are arranged in ascending order according to their burst time and the mean is calculated as quantum for every cycle according burst time of processes in ready queue.

Median Based Algorithm

Median is calculated as time quantum from the ascending order sorted queue. The median is calculated from the burst time of processes according to the number of jobs either even or odd.

3.4 Illustrations

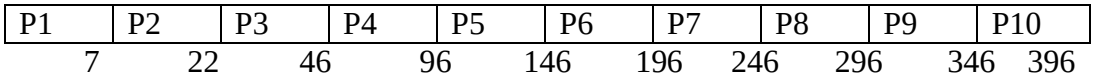
3.3.1 Case First with zero arrival time

No. of processes	Burst Time
P1	7
P2	15
P3	24
P4	84
P5	123
P6	145
P7	150
P8	175
P9	180
P10	200

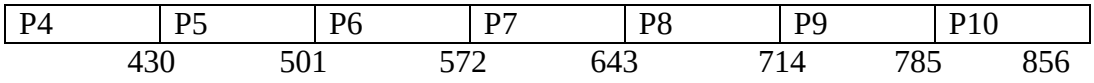
Table 3.1 Burst time in increasing order

Round Robin With varying Time Quantum (DRR)

Round First Q1= 50



Round Second Q2 = 71



Round Third Q3 = 83

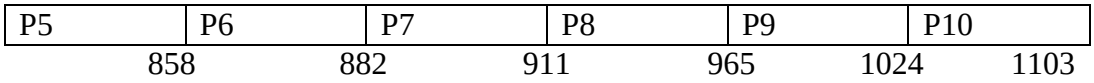


Figure 3.1 Gantt chart for DRR

Mean/average based round robin approach

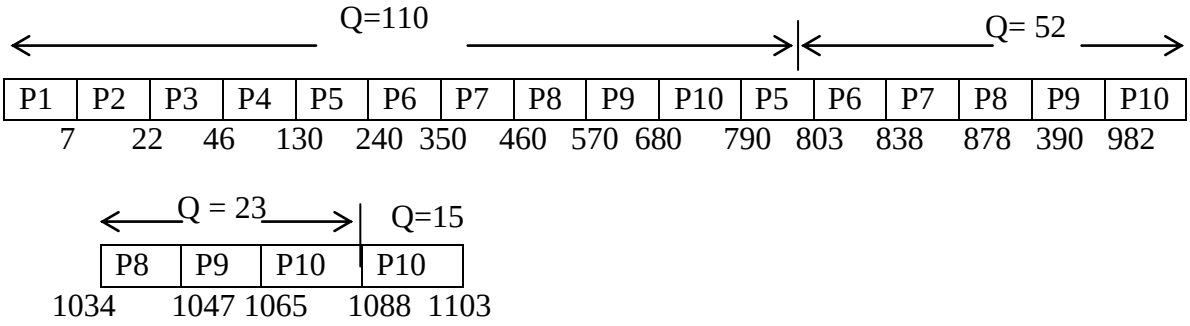


Figure 3.2 Gantt chart for Mean based RR

Median based round robin approach

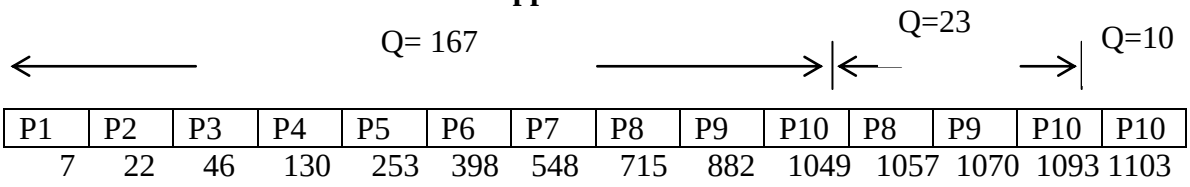


Figure 3.3 Gantt chart for Median based RR

Algorithm	QT	CS	AWT	ATAT
DRR	50,71,83	24	514.5	624.8
Mean Based	110,52,23,15	21	483.6	593.9
Median Based	167,23,10	15	353.1	463.4

Table 3.2 Comparison Table for DRR , Mean Based RR, Median Based RR

3.3.2 Case Second with zero arrival time

No of Processes	Burst Time
P1	10
P2	15
P3	35
P4	55
P5	75
P6	95
P7	115
P8	135
P9	155
P10	175

Table 3.3 Burst time in ascending order

Round Robin With varying Time Quantum (DRR)

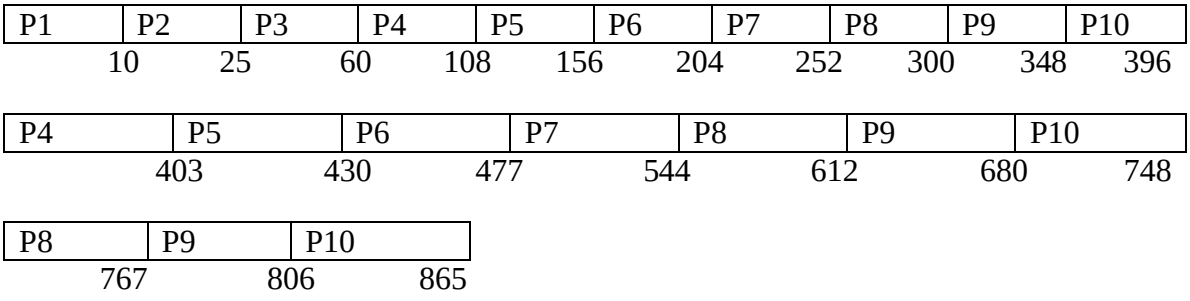


Figure 3.4 Gantt chart for RR

Mean based RR

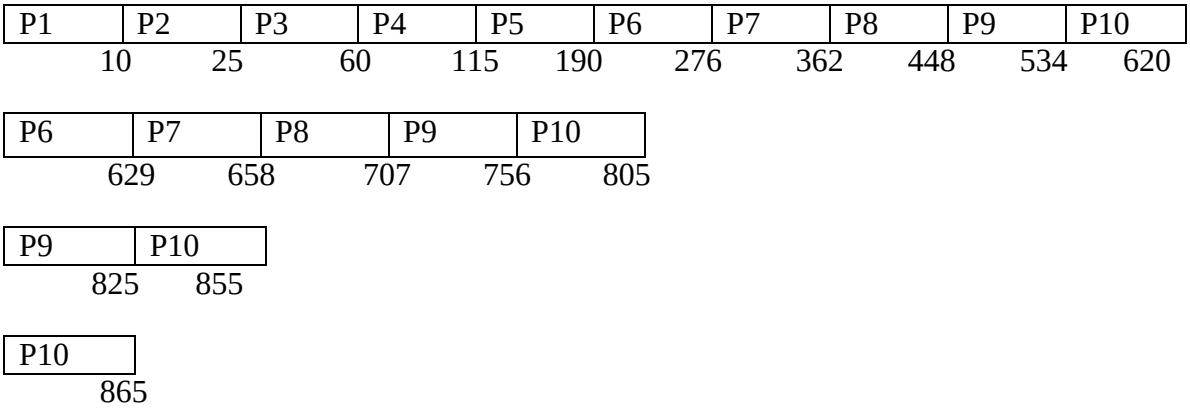


Figure 3.5: Gantt chart for Mean based

Median based RR

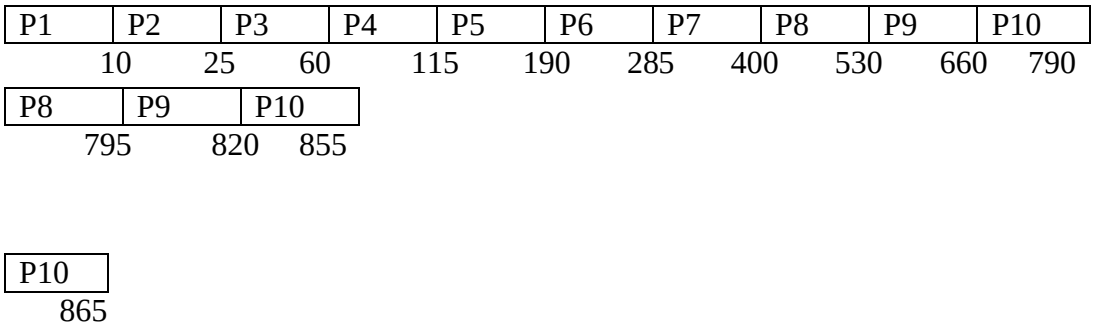


Figure 3.6 Gantt chart for median based

Algorithms	QT	CS	Awt	Atat
RR	48,68,160	20	352.2	438.7
Mean based	86,49,30,10	19	321.9	407.4
Median based	130,30,5	15	270	356.5

Table 3.4 Comparison Table ARR, Mean based RR, Median based RR.

3.3.3 Case Third with zero arrival time

No. of processes	Burst Time
P1	16
P2	28
P3	46
P4	64
P5	80
P6	120
P7	136
P8	156
P9	188
P10	220

Table 3.5 Burst time in ascending order.

Round Robin With varying Time Quantum (DRR)

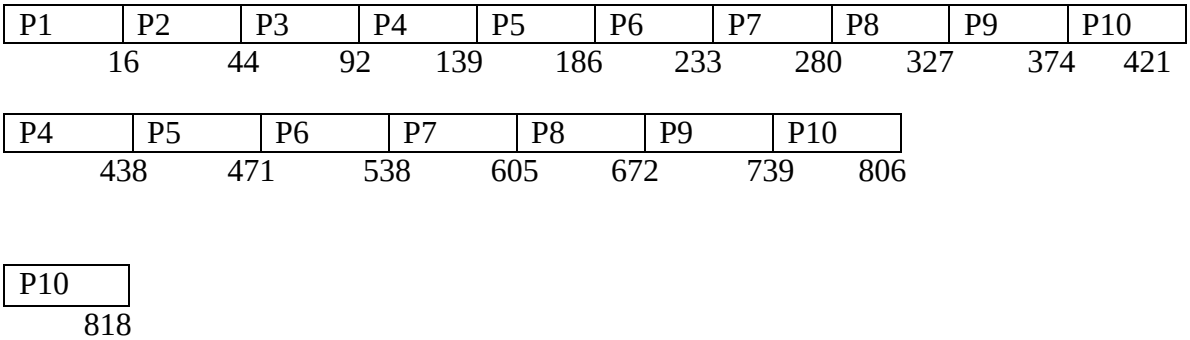


Figure3.7 Gantt chart for ARR

Mean based RR

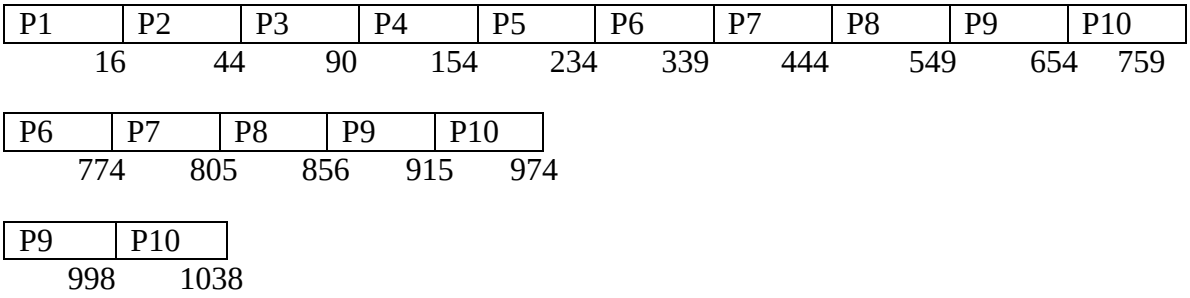


Figure 3.8 Gantt chart for Mean based. RR

Median based

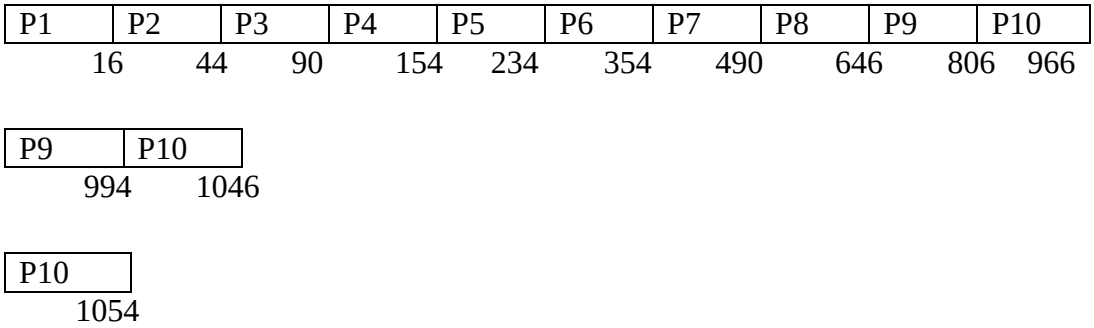


Figure 3.9 Gantt chart for RRR

Algorithms	QT	CS	Awt	Atat
DRR	47,67,470	19	361.5	443.3
Mean	105,59,40,16	19	397.1	502.5
Median	160,52,8	14	302.2	407.6

Table 3.6 Comparison Table DRR , Mean based RR , Median based RR.

4.1 Tools used

Simulation of multiprogramming environment with uniprocessor scheduling has been done using, i5 (window7, 32 bit, 4GB RAM and 2.5 GHZ processor) system.

4.1.1 Programming language

In this dissertation the algorithms are implemented in C programming language. C programming language is developed by Dennis Ritchie in 1972 A.D; this is system programming third generation language. It is power full and has flexibility as it is the choice for the most of the system programming. C was initially used for system development work, in particular the programs that make-up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be: Operating Systems, Language Compilers, Assemblers. Text Editors, Print Spoolers, Network Drivers, Modern Programs, Databases, Language Interpreters, Utilities etc.

4.1.2 Borland C++ IDE

Graphical Integrated Development Environment (IDE) is software tool that provides the platform for the programmers to make the programs or coding in convenient easier and faster. The fundamental requirement of an IDE is to provide and editor, compiler linker and debugger and simple way of to control the development of programs composed of multiple files. The Borland C++ Compiler 5.5 (BCC) is the foundation and core technology of C++Builder 5. Borland C++ Compiler 5.5 is a very fast 32-bit optimizing compiler. It includes the latest ANSI/ISO C++ language support including, the STL (Standard Template Library) framework and C++ template support and the complete Borland C/C++ Runtime Library (RTL). Also included in the free download are the Borland C/C++ command line tools such as the high performance Borland linker and resource compiler.

4.2 Data Structures used

4.2.1 Queue

Circular queue is used to represent the waiting job, ready queue. In this study, randomly generated input data are used. These random numbers represent the burst time, arrival time of processes.

4.3 Algorithms and Flow Charts

4.3.1 Pseudo code for DRR Algorithm

1.Input the no of processes with burst time

2.Schedule the jobs with simple RR with quantum from 1 to maximum burst time

3. Observe the minimum Atat for a quantum

Calculate the value of Q = n*q // for the n processes and optimized quantum q

4.For each cycle calculate the dynamic quantum q=Q/n in every cycle

5. If the ready queue is empty

6.calculate CS, Awt, and Atat

End

7.Else go to 4

End

4.3.2 Flowchart of Dynamic RR Scheduling Algorithm

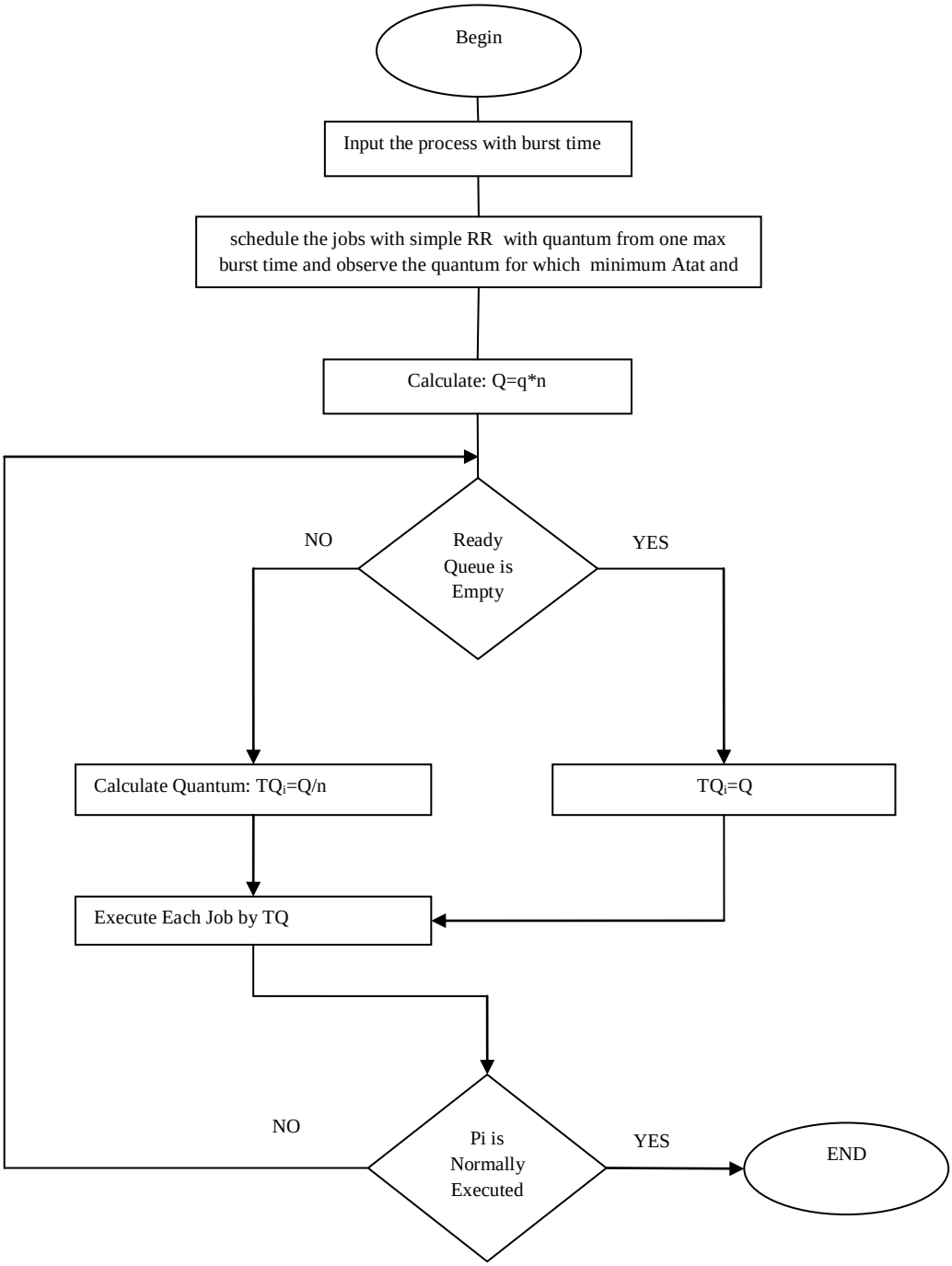


Figure 4.1 Flowchart for DRR.

4.3.3 Pseudo code for Mean based RR Algorithm

1.New process P arrives

P Enters ready queue

2.Update SR and AR

Process p is loaded from ready queue

into the CPU to be executed

3.IF (Ready Queue is Empty)

TQ = BT (p)

Update SR and AR

End if

4.IF (Ready Queue is not empty)

TQ= AVG (Sum BT of processes inready queue)

Update SR and AR

End if

5.CPU executes P by TQ time

IF (P is terminated)

Update SR and AR

End if

6.IF (P is not terminated)

Return p to the ready queue with

its updated burst time

Update SR and AR

End if

4.3.4 Flowchart of Mean Based RR algorithm

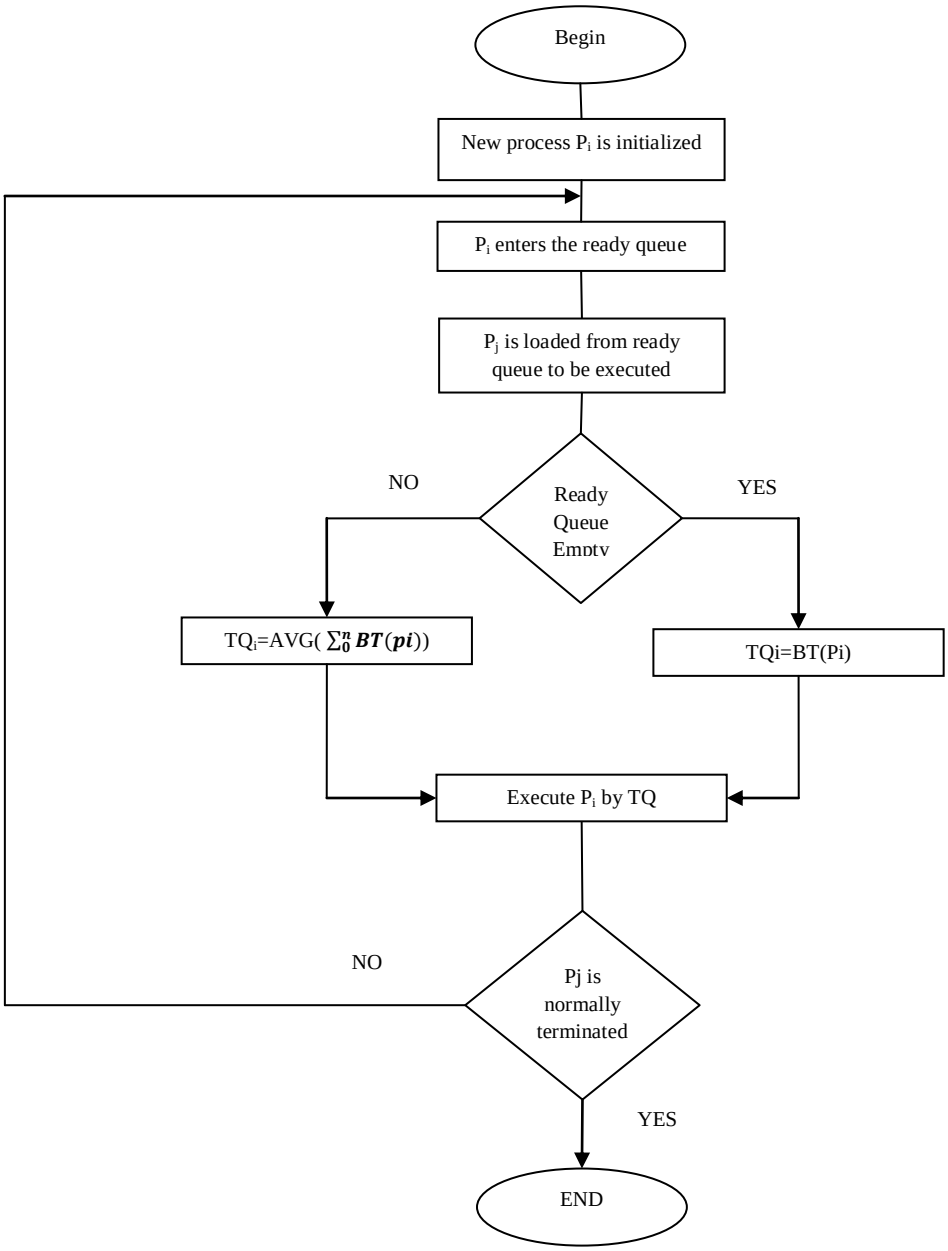


Figure 4.2 Flowchart for Mean Based RR

4.3.5 Pseudo Code for Median Based RR Algorithm

1. I/P: Process(Pn), Burst Time(bt), Arrival Time, ready queue. O/P: Context Switch(Cs), Avg.WaitingTime(Awt), Avg.Turnarround Time(Att)

2. Initialize: ready queue=0, Cs=0, Awt=0, Att=0

3. While (ready queue!=0)
// Sort the processes in ascending order according to theirBt in ready queue //Find Median //Calculate Oqt
1. //assign Oqt to each process

For each process i=1 to n
P[i]->Oqt
2. If new process arrives

Update counter and goto step3 end while
3. Cs, Awt, Att are calculated.

4. Stop and exit.

4.3.6 Flowchart of Median Based RR Algorithm

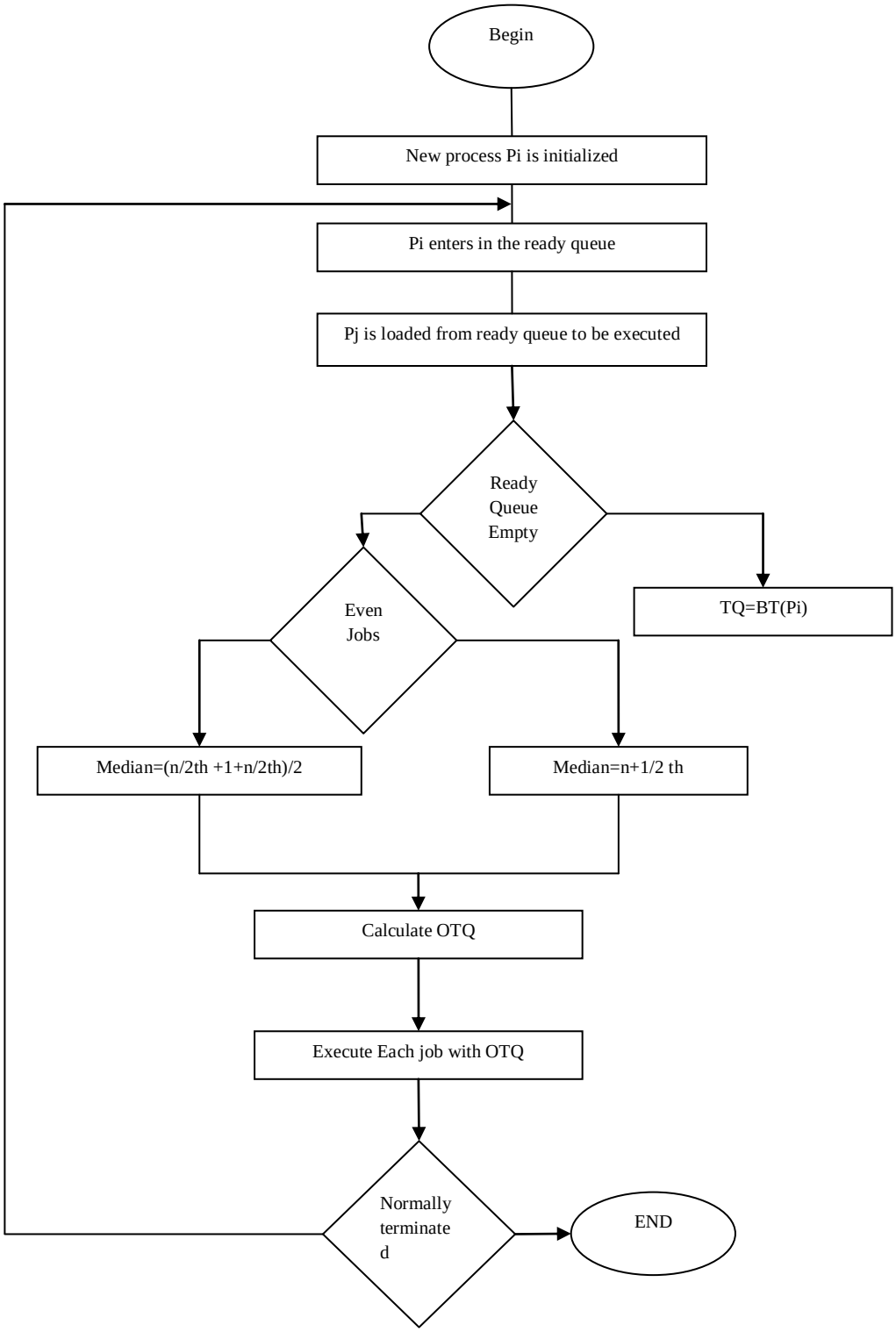


Figure 4.3 Flowchart for median based RR

CHAPTER FOUR

RESULTS ANALYSIS AND COMPARISION

5.1 Test Case Design

The algorithms are evaluated in this dissertation on three different test cases, which further categorized into three different cases such that the number of processes in each case is same but with different burst time range. Finally the algorithms are evaluated with three different test case design but with the same parameters.

5.1.1 Test Case 1:

In this test case input number of processes are varies from 20 to 200 with interval size 20 . Burst time of each process is generated using random number generator and the range of the burst time of these processes is taken in between 1 to 100 in data set 1, 100 to 500 in data set 2, and 500 to 1000 in data set 3.

5.1.2 Test Case 2:

In this test case input number of processes are varies from 50 to 500 with interval size 50 . Burst time of each process is generated using random number generator and the range of the burst time of these processes is taken in between 1 to 100 in data set 1, 100 to 500 in data set 2, and 500 to 1000 in data set 3.

5.1.3 Test Case 3:

In this test case input number of processes are varies from 100 to 1000 with interval size 100 . Burst time of each process is generated using random number generator and the range of the burst time of these processes is taken in between 1 to 100 in data set 1, 100 to 500 in data set 2, and 500 to 1000 in data set 3.

5.2 Data Collection and Analysis:

5.2.1 Table and Graph for Case 1 :

No of Processes	DRR		Mean		Median	
	Awt	Atat	Awt	Atat	Awt	Atat
20	507.5	560.35	498.5	551.35	482.75	535.59
40	1085.05	1138.07	1063.25	1116.27	1001.59	1054.62
60	1566.69	1623.44	1679.98	1754.73	1615.27	1672.00
80	2176.45	2231	2128	2182.68	2078.05	2132.63
100	2648.17	2701.20	2571.84	2570.88	2514.32	2567.36
120	2940.90	2989.80	2738.62	2787.52	2698.44	2747.35
140	3883.84	3937.40	3576.43	3629.99	3483.50	3537.06
160	4166.22	4218.26	3998.84	4050.89	3982.97	4035.01
180	4438.73	4487.68	4066.25	4124.20	4075.85	4115.80
200	5207.06	5258.96	5010.96	5062.85	4945.25	4997.14

Table 5.1 Input Processes are taken in 20 to 200 and their burst time range in between (1 to 100)

No of Processes	DRR		Mean		Median	
	Awt	Atat	Awt	Atat	Awt	Atat
20	2800.70	3082.30	2799.85	3080.45	2787.60	3068.20
40	6262.13	6560.00	6224.15	6522.02	6065.17	6363.05
60	9274.18	9588.77	9220.57	10016.15	9050.67	10000.25
80	11952.38	12234.03	11276.38	11558.03	11240.06	11521.71
100	14132.81	14413.62	13774.43	14055.24	13720.03	13552.84
120	18760.67	18307.03	18601.82	18696.16	18465.80	18560.14
140	21698.59	21999.69	21057.00	21358.11	20612.81	20913.91
160	26918.68	26231.94	25555.50	25868.76	25354.21	25667.46
180	27929.61	28228.85	27368.52	27667.76	26196.21	26494.45
200	30134.71	30426.33	29919.57	30211.20	28409.63	28701.25

Table 5.2 Input Processes are taken in 20 to 200 and their burst time range in between (100 to 500)

No of Processes	DRR		Mean		Median	
	Awt	Atat	Awt	Atat	Awt	Atat
20	8534.25	9294.25	8187.90	8947.90	7533.75	8293.75
40	16297.60	17030.47	16286.78	17019.65	14069.72	14802.60
60	24969.00	25710.50	24961.63	25703.13	21201.25	21942.75
80	35100.76	30749.49	35064.86	35835.59	34929.14	35699.86
100	43018.96	43774.80	42693.94	43449.78	37172.19	37928.03
120	49906.35	50638.58	49549.88	50282.12	43143.68	43874.91
140	59427.62	60173.85	58625.08	59370.31	51621.91	52366.14
160	66535.71	67207.31	64187.34	64918.94	59108.48	59840.08
180	75874.11	76611.89	74480.91	75218.68	66400.13	67137.90
200	88383.09	89150.71	87911.60	88678.22	77105.63	77871.25

Table 5.3 Input Processes are taken in 20 to 200 and their burst time range in between (500 to 1000)

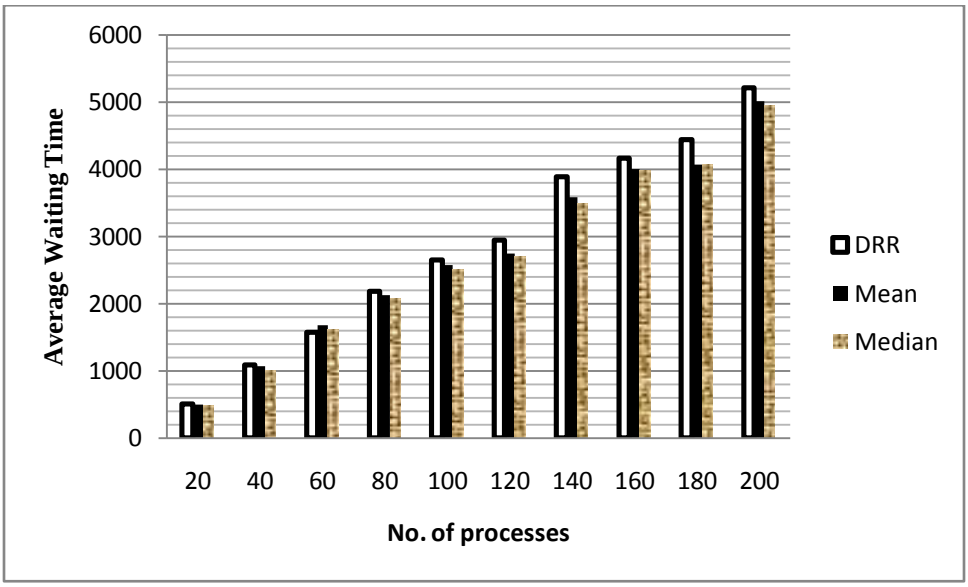


Fig. 5.1: Graph for Average Waiting time (Table 5.1)

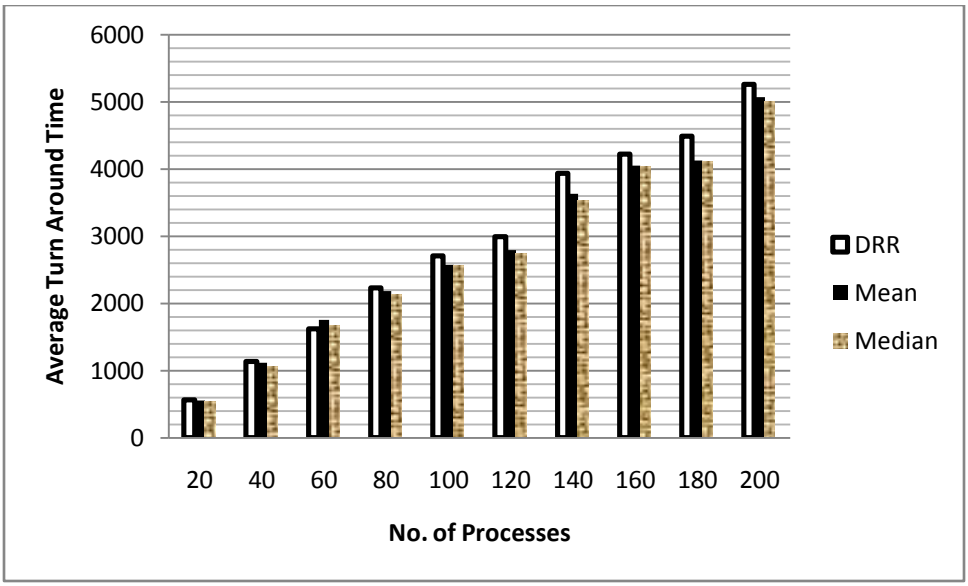


Fig. 5.2 : Graph for Average Turnaround time (Table 5.1)

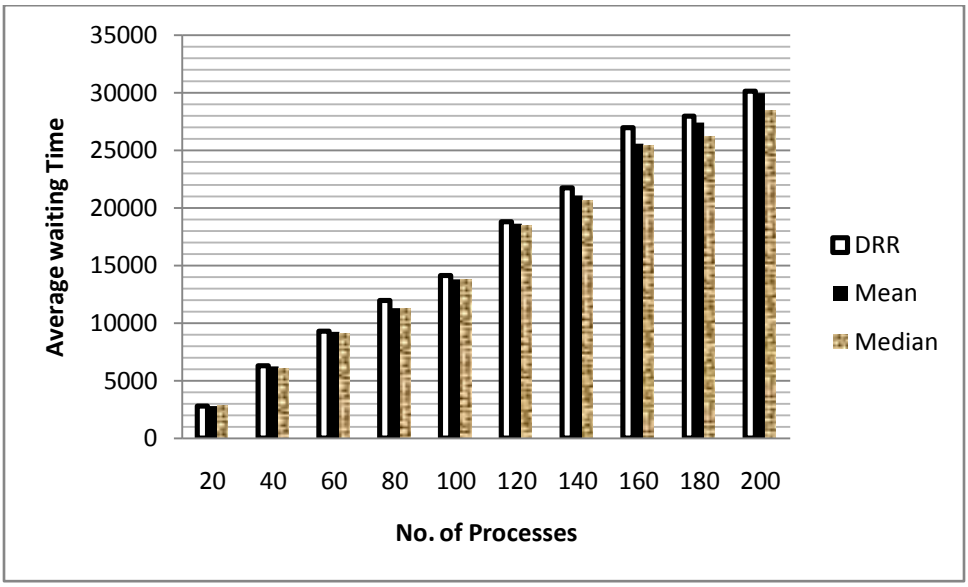


Fig. 5.3: Graph for Average Waiting time (Table 5.2)

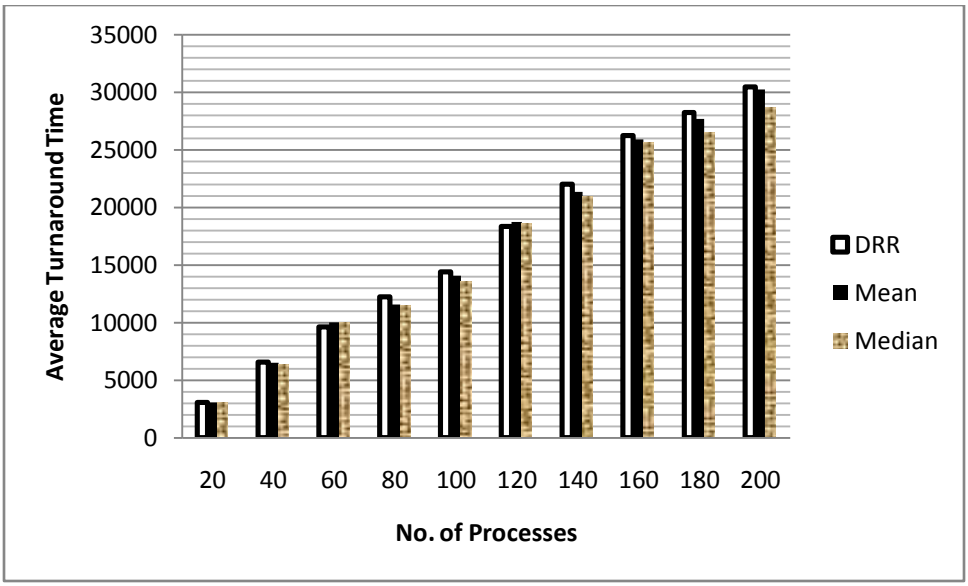


Fig. 5.4 : Graph for Average Turnaround time (Table 5.2)

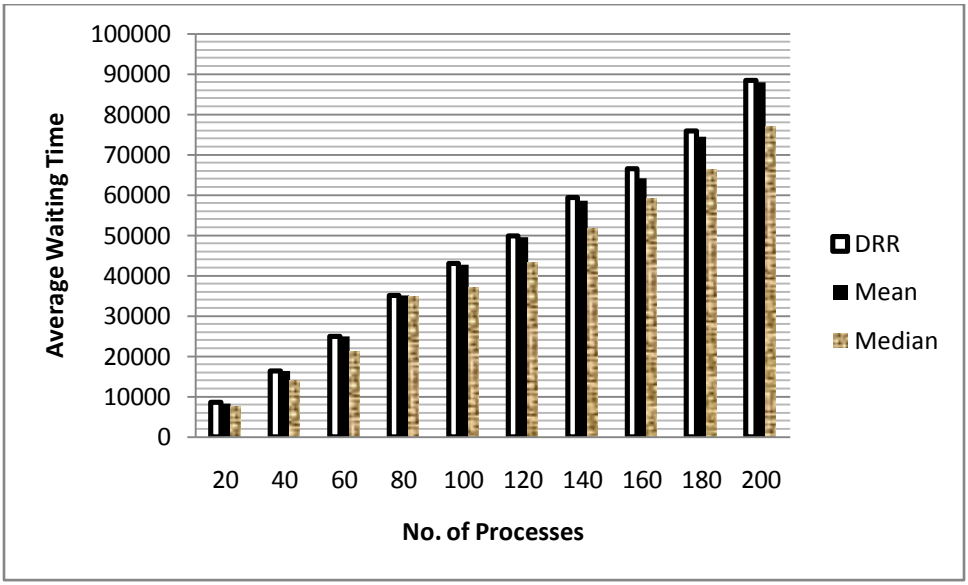


Fig. 5.5: Graph for Average Waiting time (Table 5.3)

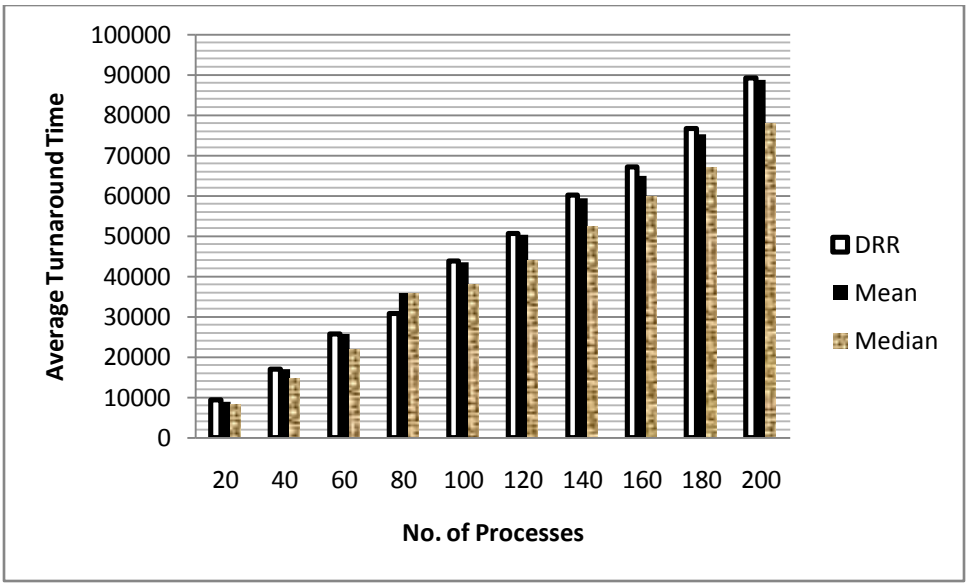


Fig. 5.6 : Graph for Average Turnaround time (Table 5.3)

In the above graphs and table(5.1 to 5.3) the average waiting time of Median based RR is 16.07%,6.62% and 17.77% lower and average turnaround time is 11.32%,6.55% and 17.17% lower respectively than DRR. Similarly if we compare the performance of Median Based RR it seems average waiting time is 6.16%,5.32% and 17.74% less and average turnaround time 5.85%,5.26% and 17.14%less respectively than Mean based RR.

5.2.2 Table and Graph for Case 2:

No of Processes	DRR		Mean		Median	
	Awt	Atat	Awt	Atat	Awt	Atat
50	7545.68	7629.34	7169.30	7453.96	7128.78	7411.44
100	15703.45	15595.23	15268.15	15559.23	14898.30	15190.08
150	26063.41	26375.50	25623.03	25935.12	24505.85	24817.95
200	33733.27	40032.23	32299.46	32598.41	31597.76	31896.72
250	40750.12	40955.98	40000.32	40306.19	39969.83	40275.70
300	48287.33	47987.61	47623.94	47924.22	47412.96	47713.24
350	55973.09	55271.18	54682.56	54980.65	53406.90	53704.99
400	65876.21	66179.79	65610.64	65914.23	64154.87	64458.45
450	67956.90	68247.23	67861.79	68152.12	67555.94	67846.27
500	79696.01	78994.55	78333.45	78631.99	78176.07	78474.62

Table 5.4 Input Processes are taken in 50 to 500 and their burst time range in between (100 to 500)

No of Processes	DRR		Mean		Median	
	Awt	Atat	Awt	Atat	Awt	Atat
50	21461.90	22213.28	21082.40	21833.78	18494.18	19245.56
100	42033.60	42772.15	41667.14	42405.69	36187.32	36925.87
150	65186.57	65934.65	63880.80	64628.88	56296.38	57044.45
200	85577.43	86317.41	84809.45	85549.42	73444.60	74184.58
250	107679.12	108434.34	107702.79	108458.02	94942.50	95697.73
300	127790.42	128534.88	127784.32	128528.77	111997.52	112741.98
350	148090.78	148834.06	148049.70	148792.98	130198.17	130941.45
400	170734.06	171480.09	169639.03	170385.06	149727.42	150473.44
450	191261.69	192004.55	191182.67	191925.53	166074.20	166817.06
500	213845.98	214593.28	211574.53	212321.83	185433.42	186180.72

Table 5.5 Input Processes are taken in 50 to 500 and their burst time range in between (500 to 1000)

No of Processes	DRR		Mean		Median	
	Awt	Atat	Awt	Atat	Awt	Atat
50	13337.76	13886.62	13147.92	13696.78	13061.30	13610.16
100	26383.84	26925.86	28832.34	29374.35	26815.21	27357.22
150	35794.40	36245.07	29542.06	29992.74	30069.19	30519.87
200	56033.85	56588.01	54674.14	55228.30	53883.84	54437.99
250	62659.79	63160.24	58214.60	58715.05	57998.10	58498.55
300	75735.16	76238.65	72159.14	72662.63	71914.54	72418.02
350	90839.77	91353.09	85637.95	86151.28	85485.88	85999.20
400	96365.73	96850.55	89969.06	90453.89	90252.60	90737.43
450	116714.60	117241.95	117589.74	118117.08	115534.93	116062.27
500	118241.03	118718.45	108967.80	109445.22	110997.54	111474.95

Table 5.6 Input Processes are taken in 50 to 500 and their burst time range in between (1 to 1000)

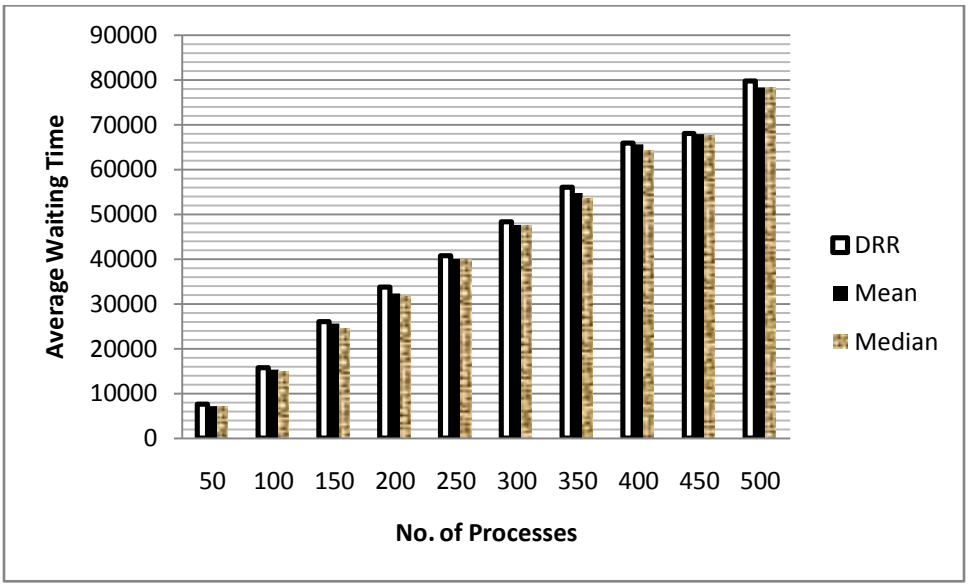


Fig. 5.7: Graph for Average Waiting time (Table 5.4)

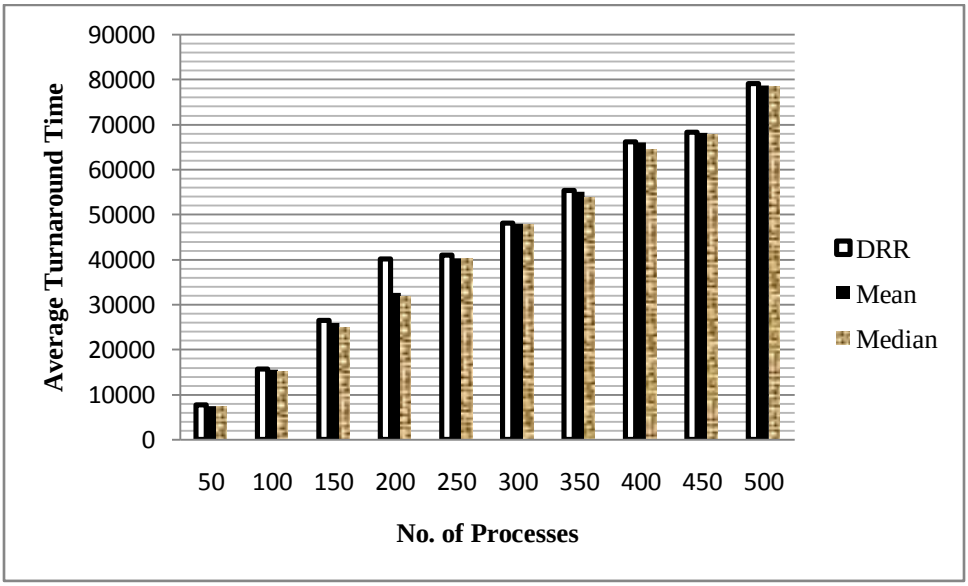


Fig. 5.8 : Graph for Average Turnaround time (Table 5.4)

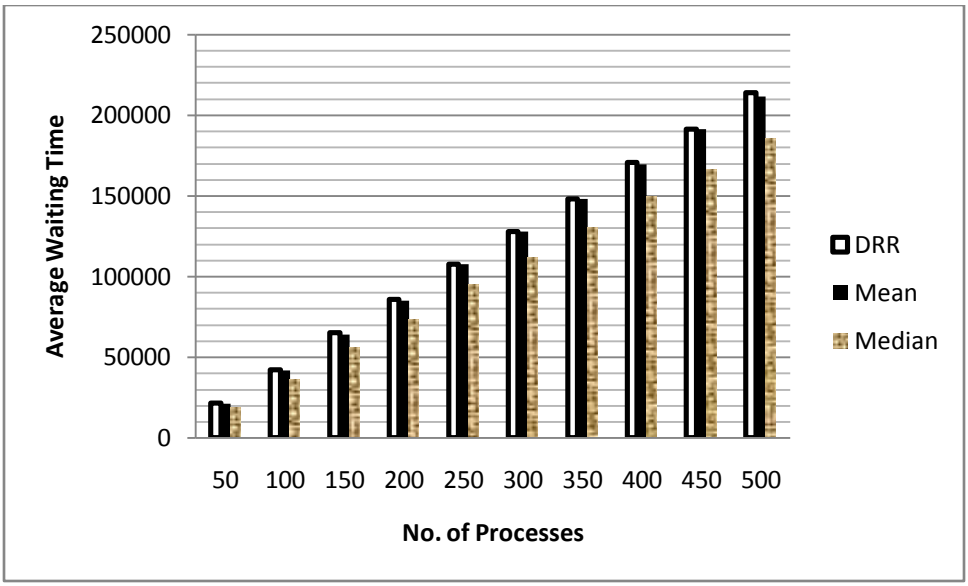


Fig. 5.9: Graph for Average Waiting time (Table 5.5)

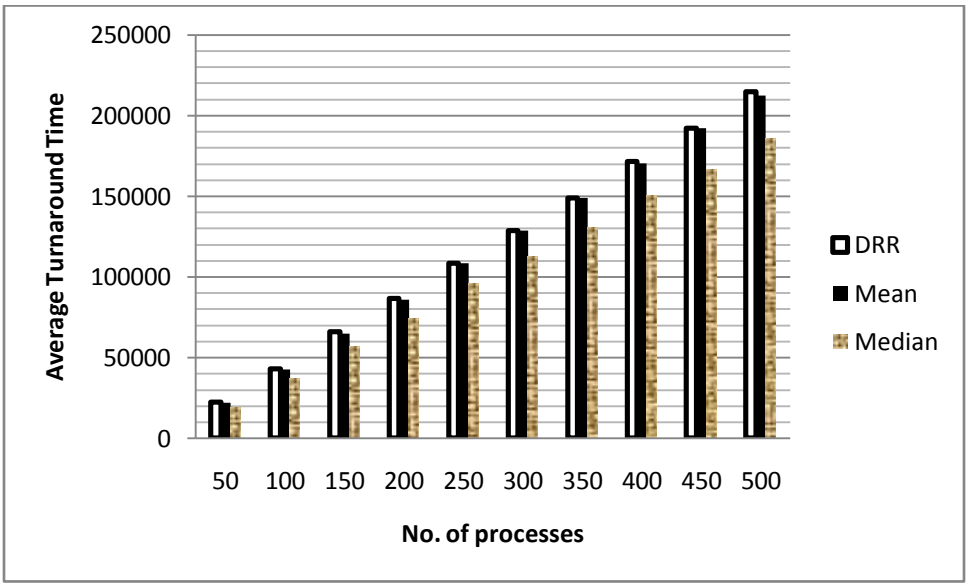


Fig. 5.10 : Graph for Average Turnaround time (Table 5.5)

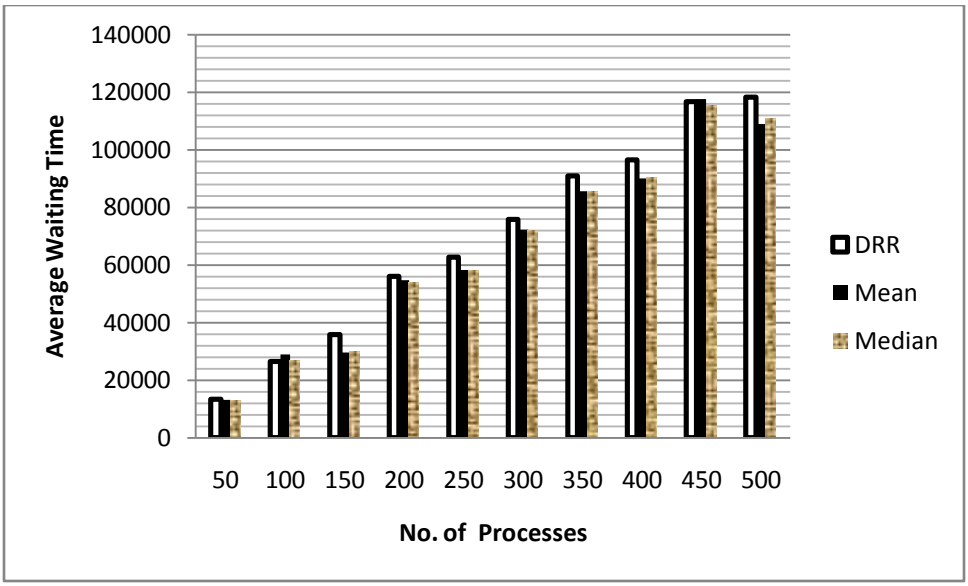


Fig. 5.11: Graph for Average Waiting time (Table 5.6)

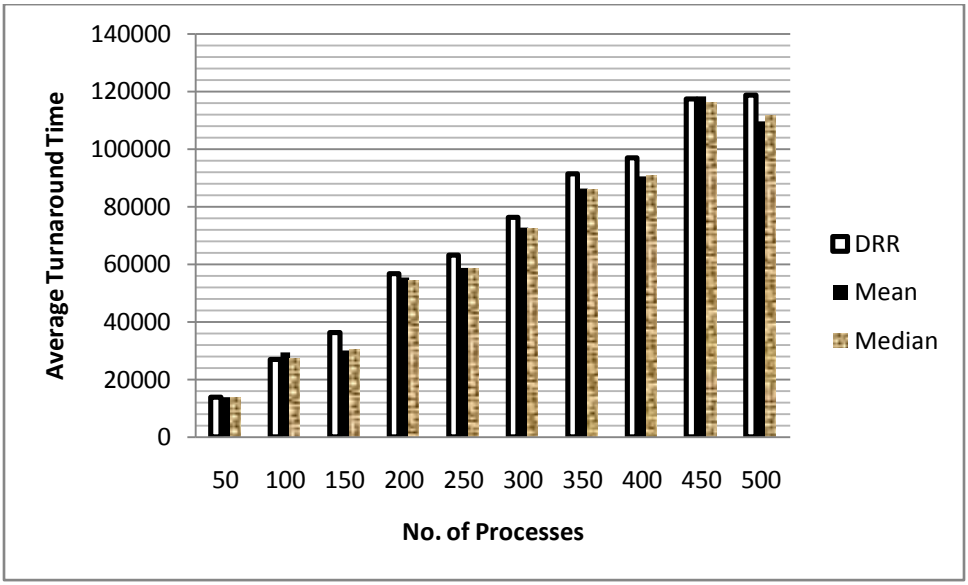


Fig. 5.12 : Graph for Average Turnaround time (Table 5.6)

In the above graphs and table (5.4 to 5.6)) the average waiting time of Median based RR is 5.98%,16.79% and 19.04% lower and average turnaround time is 25.50%,16.35% and 18.76% lower respectively than DRR. Similarly if we compare the performance of Median Based RR it seems average waiting time is 4.56%,15.47% and 7.52% less and average turnaround time 2.43%,15.32% and 1.82%less respectively than Mean based RR

5.2.3 Table and Graph for Case 3:

No of Processes	DRR		Mean		Median	
	Awt	Atat	Awt	Atat	Awt	Atat
100	14732.17	15017.42	14631.35	14916.61	13671.34	13956.60
200	32252.28	32554.29	31634.88	31936.89	29326.00	29628.03
300	46662.61	46971.37	49431.60	49740.37	49214.84	49523.60
400	59740.18	60033.80	60795.72	61089.34	61465.36	61758.98
500	75235.56	75525.58	74174.49	74464.51	72609.03	72899.04
600	92125.98	92421.28	90717.13	91012.43	87673.34	87968.65
700	108238.39	108535.80	107285.89	107583.31	104111.00	104408.42
800	123759.70	124056.45	123154.31	123451.05	116728.84	117025.58
900	143742.03	144042.67	142278.89	142579.53	134953.61	135254.25
1000	159744.42	160048.63	159480.78	159784.97	152814.97	153119.16

Table 5.7 Input Processes are taken in 100 to 1000 and their burst time range in between (100 to 500)

No of Processes	DRR		Mean		Median	
	Awt	Atat	Awt	Atat	Awt	Atat
100	42823.78	43573.17	42537.51	43285.90	37412.94	38161.33
200	86495.30	87249.23	86119.43	86873.35	74850.84	75604.75
300	129920.41	130668.75	128715.88	129464.23	112051.32	112799.67
400	174048.86	174797.93	172004.55	172753.63	150061.84	150810.92
500	220635.30	221393.53	217359.64	218117.88	189917.84	190676.08
600	258769.61	259522.75	258369.45	259122.59	224465.48	225218.63
700	300115.03	300862.90	299007.00	299754.88	259895.02	260642.87
800	344295.00	345044.38	342507.97	343257.34	301084.69	301834.06
900	392133.22	392885.16	388896.31	389648.25	337258.28	338010.25
1000	427148.81	427895.13	426863.59	427609.94	373997.44	374743.78

Table 5.8 Input Processes are taken in 100 to 1000 and their burst time range in between (500 to 1000)

No of Processes	DRR		Mean		Median	
	Awt	Atat	Awt	Atat	Awt	Atat
100	23560.59	24359.35	24331.02	24229.78	23836.63	24300.39
200	47852.82	48345.80	45257.93	45750.91	45553.16	46046.14
300	72861.43	73343.58	67088.23	67570.38	66788.02	67270.17
400	99906.94	100401.22	94207.89	94702.17	93712.46	94206.74
500	126501.07	127008.91	121749.28	122357.12	121801.74	122209.59
600	146118.53	146610.25	139181.16	139672.88	138871.48	139363.22
700	169941.80	170433.94	160526.80	162018.94	161760.67	161252.81
800	199633.77	200131.41	188111.83	189609.47	189136.23	189533.88
900	223429.30	223927.39	211630.45	212228.55	211738.50	212136.59
1000	247741.41	248238.48	231916.98	232414.05	233371.73	232368.80

Table 5.9 Input Processes are taken in 100 to 1000 and their burst time range in between (1 to 1000)

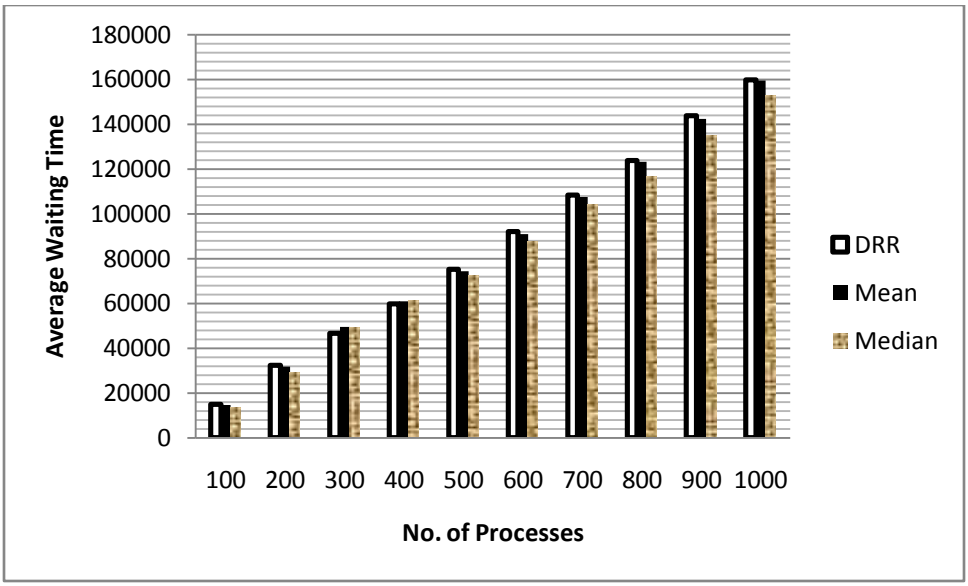


Fig. 5.13: Graph for Average Waiting time (Table 5.7)

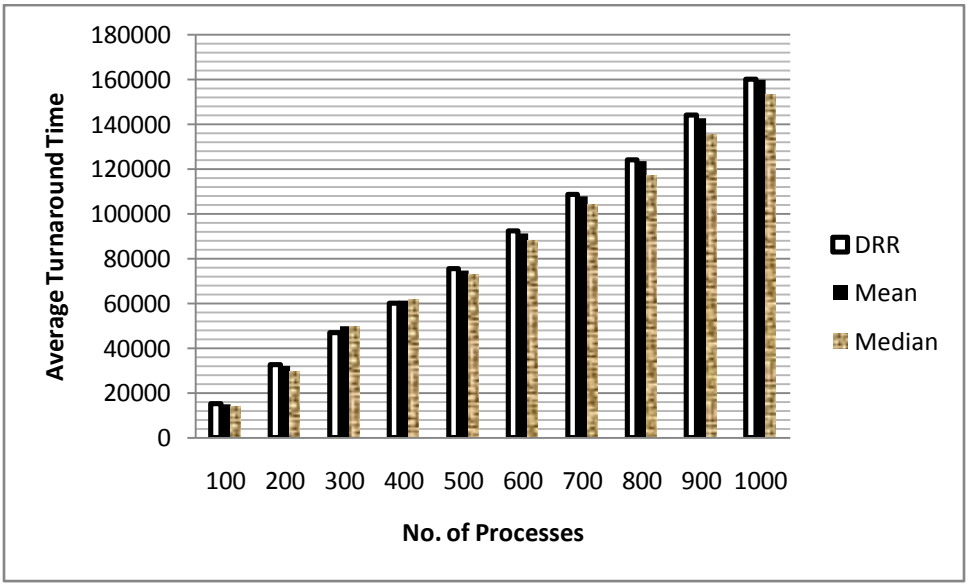


Fig. 5.14 : Graph for Average Turnaround time (Table 5.7)

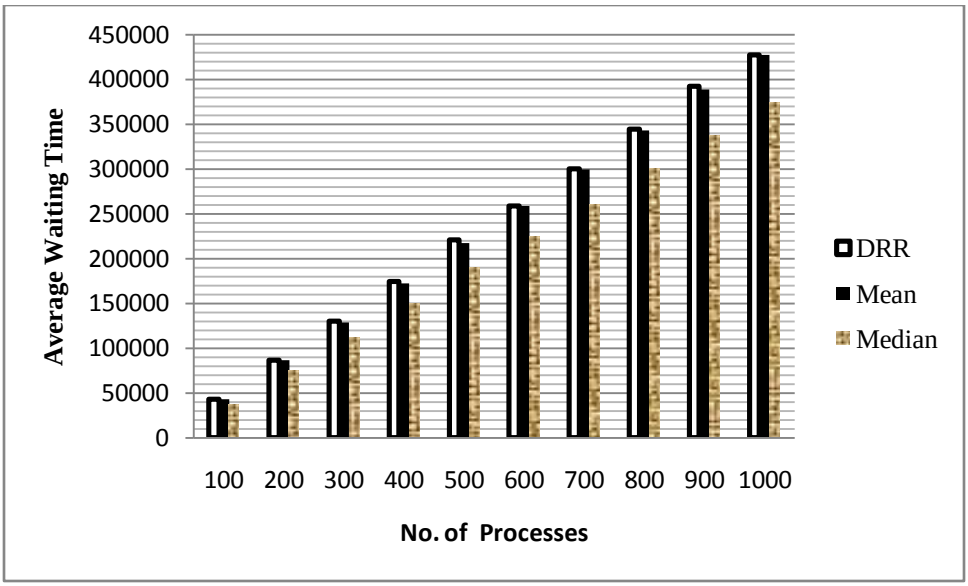


Fig. 5.15: Graph for Average Waiting time (Table 5.8)

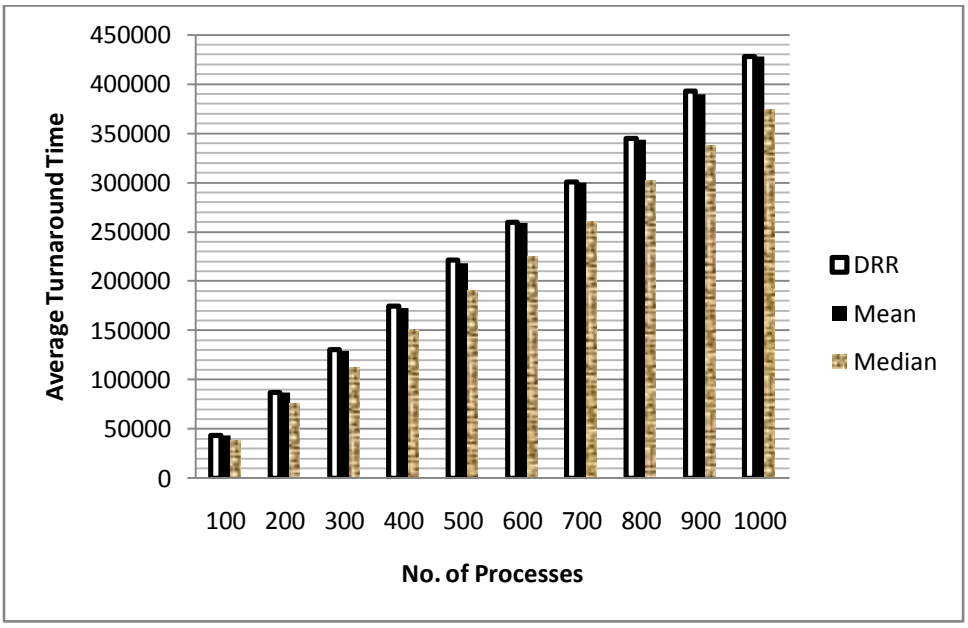


Fig. 5.16 : Graph for Average Turnaround time (Table 5.8)

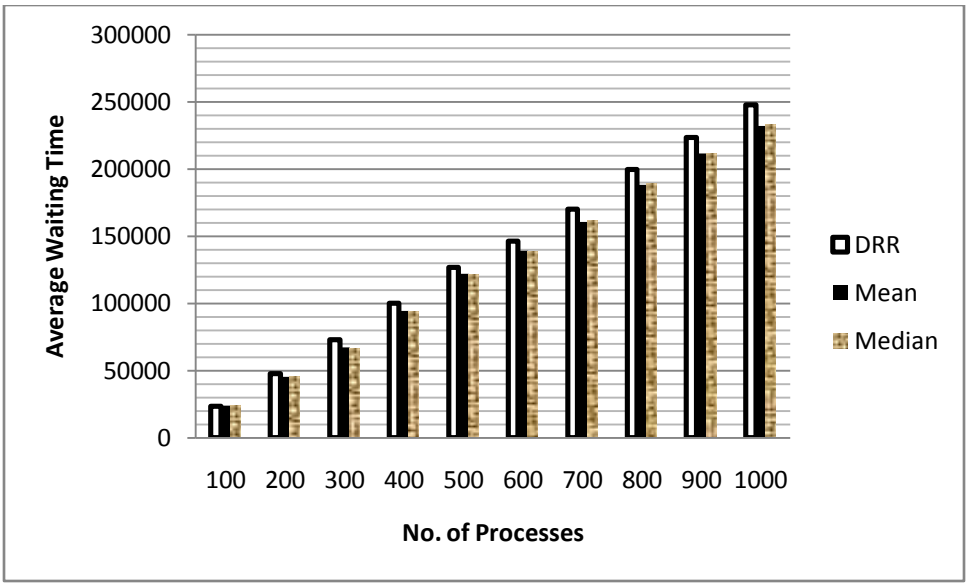


Fig. 5.17: Graph for Average Waiting time (Table 5.9)

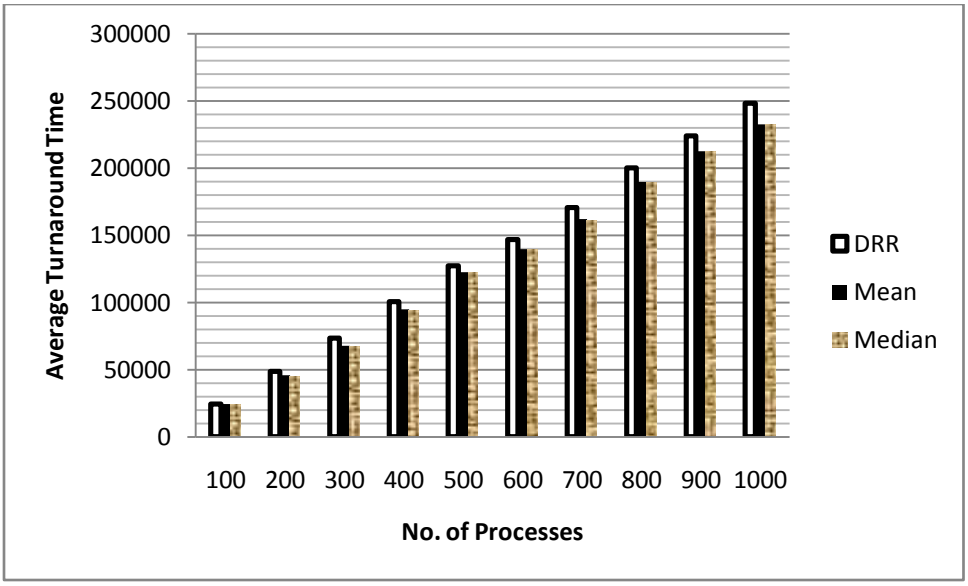


Fig. 5.18 : Graph for Average Turnaround time (Table 5.9)

In the above graphs and table (5.7 to 5.9)) the average waiting time of Median based RR is 7.76%,16.27% and 9.09% lower and average turnaround time is 9.88%,16.23% and 9.03% lower respectively than DRR. Similarly if we compare the performance of Median Based RR it seems average waiting time is 7.02%,15.31% and 2.07% less and average turnaround time 7.79%,15.27% and 0.64%less respectively than Mean based RR

CHAPTER FIVE

CONCLUSION

6.1 Conclusion

The optimal quantum is the central concern for the Round Robin scheduling algorithm in its applications. The large quantum implies the algorithm to be FIFO and small quantum gives the overhead of context switch, so many algorithms has been designed and claimed for the best better algorithm but regarding my analysis I have observed three algorithms namely DRR, Mean based RR and Median based RR, on different parameters for three different cases, and Median based RR algorithm is the best among them.

Comparing the performance of median based RR , DRR and mean based RR, the average waiting time of median based RR is 5.98% to 19.04% less than DRR and average turnaround time is 6.55% to 25.50% less. Similarly average waiting time of median based RR is 2.07% to 17.74% less than mean based RR similarly average turnaround time is 0.64% to 17.14% less.

CHAPTER SIX

RECOMMENDATION AND LIMITATIONS

7.1 Recommendation

This analysis study is based on adopting the optimal quantum size. This research study includes the performance evaluation of the three variants of dynamic round robin scheduling algorithms (namely DRR, Mean based RR and Median based RR) based on different parameters. Further comparative analysis can be done with other central values like quartiles, deviation , standard deviation value.

7.2 Limitations

In this study randomly generated data set are used under some criteria. The standard data set are not used , result might some different under standard data set. The optimization of the source code might be done which may enhance the performance and efficiency.

REFERENCES:

[1] Rami Abielmona, Scheduling Algorithmic Research, Department of Electrical and Computer Engineering Ottawa-Carleton Institute, 2000.

[2] William Stallings Internals and Design Principles, Fourth Edition, Operating Systems (P 394).

[3] Silber Schatz, Galvin, Gangne, Operating System Concepts, 8thEdition (Page 194).

[4]Noon Abbas,Kalakech Ali, KadrySeifedine, *A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average*, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No.1, May 2011.

[5] NayakDebashree, MallaSanjeev Kumar, DebadarshiniDebashree, *Improved Round Robin Scheduling using Dynamic Time Quantum*, International Journal of Computer Applications(0975-8887) Volume 38-No.5, January 2012.

[6] GhishingAshim, Analysis of the Varying Time Quantum Round Robin Scheduling; CDCSIT, TU.

[7] Panda Sanjaya Kumar, Dash Debasis, Rout Jitendra Kumar, *A Group based Time Quantum Round Robin Algorithm using Min-Max Spread Measure*, International Journal of Computer Applications (0975 - 8887) Volume 64- No.10, February 2013.

[8] Rami J. Matharneh, Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the now Running Processes, Department of Management Information Systems, American Journal of Applied Sciences 6 (10):1831-1837, 2009, ISSN 1546-9239.

[9] Pallabbenerjee, probalbenerjee, shwetasonali dhal, Comparative performance analysis of average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm using static Time Quantum, IJITEE, ISSN:2278-3075, Volume – 1, Issue-3, August 2012.

[10] S.M. Mostafa, S.Z. Rida, and S.H. Hamad, Finding time quantum of Round Robin CPU scheduling in general computing systems using integer programming, International Journal of Research and Review in Applied Science. 5(1), 64-71.2010.

[11] P. SurendraVarma, A Best possible Time Quantum for Improving Shortest Remaining Burst Round Robin (SRBRR) Algorithm, International journal of Advance Research in Computer Science and Software Engineering, Volume- 2, Issue-11, November 2012.

[12] ShahramSaeidi, HakimehAlemiBaktash, Determining the Optimum Time Quantum Value in Round Robin Process Scheduling Method, Information Technology of Computer Science, 2012,10,67-73.

[13] H.S. Behera, R. Mohanty, S. Sahu, and S.K. Bhoi, Comparative performance analysis of Multi-Dynamic Time Quantum Round Robin (MDTQRR) algorithm with arrival time, Indian Journal of Science and Engineering , 2(2), 262-271, 2011.

[14] J. Nieh, Ch. Vaill, and H. Zhong, Virtual-Time Round-Robin: An O(1) Proportional Share Schedular. Proceedng of the 2001 USENIX Annual Technical Conference, USA, 2001.

[15] M.H. Zahedi, M. Ghazizadeh, and M. Naghibzadeh, Fuzzy Round Robin CPU Scheduling (FRRCS) Algorithm, Advances in Computer and Information Science and Engineering. 348-353,2008.

[16] C. Yaashuwanth, and R. Ramesh, Intelligent Time Slice for Round Robin in Real Time Operating Systems, IJRRAS, 2(2),126-131, 2010.

[17] M.U. Siregar, A New Approach to CPU Scheduling Algorithm: Genetic Round Robin, International Journal of Computer Applications, Vol. 47,No.19., 2012.