ISSN No. 0976-5697

Volume 3, No. 5, Sept-Oct 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Improved Shortest Remaining Burst Round Robin (ISRBRR) Using Optimal Time quantum

P.Surendra Varma
Dept. Name of Computer Science & Engineering
NRI Institute of Technology
Vijayawada, India
surendravarma008@gmail.com

Abstract: Round Robin (RR) performs optimally in timeshared systems because each process is given an equal amount of static time quantum. But the effectiveness of RR algorithm solely depends upon the choice of time quantum. I have made a comprehensive study and analysis of RR algorithm and SRBRR algorithm. I have proposed an improved version of SRBRR (Shortest Remaining Burst Round Robin) by assigning the processor to processes with shortest remaining burst in round robin manner using the optimal time quantum. Time quantum is computed with the help of median and highest burst time.

Keywords: Operating System, Scheduling Algorithm, Round Robin, Context switch, optimal tq

I. INTRODUCTION

A. Any CPU scheduling algorithm relies on the following criteria. They are:

a. CPU utilization:

We want to keep the CPU as busy as possible that means CPU is not free during the execution of processes. Conceptually the CPU utilization can range from 0 to 100 percent.

b. Response time:

Response time is the time from the submission of a request until the first response is produced.

c. Throughput:

One measure work is the number of processes that are completed per time unit that means the number of tasks per second which the scheduler manages to complete the tasks.

d. Turnaround Time:

The time interval from the time of submission of a process to the time of completion is the turnaround time. Total turnaround time is calculation is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

e. Waiting Time:

The waiting time is not the measurement of time when a process executes or does I/O completion; it affects only the amount of time of submission of a process spends waiting in the ready queue. We keep average waiting time should be less.

II. PRELIMINARIES

A process is an instance of a computer program that is being executed. The processes waiting to be assigned to a processor are put in a queue called *ready queue*. The time for which a process holds the CPU is known as *burst time*. *Arrival Time* is the time at which a process arrives at the ready queue. The interval from the time of submission of a process to the time of completion is the turnaround time.. *Waiting time* is the amount of time a process has been waiting in the ready queue. The number of times CPU switches from one process to another is known as *context switch*. The optimal scheduling algorithm will have minimum waiting time, minimum turnaround time and minimum number of context switches.

A. Basic Scheduling Algorithms:

a. First Come First Serve (FCFS):

In the First-Come-First-Serve (FCFS) algorithm, the CPU is assigned immediately to that process which arrives first at the ready queue. Processes are dispatched according to their arrival time on the ready queue. Being a non preemptive discipline, once a process has a CPU, it runs to completion.

b. Shortest Job First (SJF):

In this strategy the scheduler arranges processes with the Burst times in the ready queue, so that the process with low burst time is scheduled first.
If two processes having same burst time and arrival time, then FCFS procedure is followed.

c. Shortest Remaining Time First (SRTF):

This is same as the SJF with pre emption, which small modification. For scheduling the jobs system need to consider the remaining burst time of the job which is presently executed by the CPU also along with the burst time of the jobs present in the ready queue.

d. Priority Scheduling Algorithm:

It provides the priority to each process and selects the highest priority process from the ready queue.

e. Round robin Scheduling Algorithm:

Round Robin (RR) is one of the oldest, simplest, and fairest and most widely used scheduling algorithms, designed especially for time-sharing systems. Here every

process has equal priority and is given a time quantum after which the process is preempted.

III. RELATED WORK

Efforts have been made to modify SRBRR [3] in order to give better turnaround time, average waiting time and minimize context switches.

IV. PROPOSED ALGORITHM

A. The proposed algorithm works as follows:

a. All the processes present in ready queue are sorted in ascending order.
b. While (ready queue!= NULL)
   TQ = Ceil ((Highest B.T + median)/ 2)
c. Assign TQ to process
   Pi ->TQ
d. If (i<n) then go to step 3
e. If a new process is arrived,
   Update the counter n and
   go to step1
   End of while
f. Average waiting time, average turnaround time and Number of context switches are calculated.
g. End

V. EXPERIMENTS & RESULTS

A. Assumptions:

All experiments are assumed to be performed in uniprocessor environment and all the processes are independent from each other. Attributes like burst time and priority are known prior to submission of process. All processes are CPU bound. No process is I/O bound. Processes with same arrival time are scheduled.

B. Illustration and Results:

Case-I:

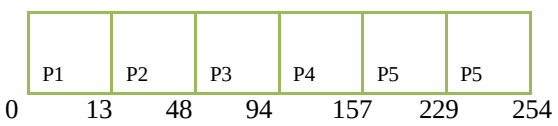
Let us assume five processes, with increasing burst time (P1 = 13, P2 = 35, P 3 = 46, P4 = 63, p5= 97) as shown in TABLE.

Table: 1

Process	Burst Time
P1	13
P2	35
P3	46
P4	63
P5	97

Now, as per the algorithm Time Quantum is calculated as follows

TQ = Ceil((Highest B.T + median)/ 2)
TQ = Ceil ((97 + 46) / 2) = 72



Number of Context Switches = 5
Average Waiting Time = (0+13+48+94+157) / 5 = 62.4
Average Turnaround Time = (13+48+94+157+254) / 5
= 113.2

Table 1: Comparison between RR, SRBRR and Proposed algorithm
(case – I)

Table: 2

Algorithm	Time Quantum	Avg.TAT	Avg.WT	CS
RR	**25**	**148.2**	**97.4**	**11**
SRBRR	**46**	**122.4**	**71.6**	**7**
ISRBRR	**72**	**113.2**	**62.4**	**5**

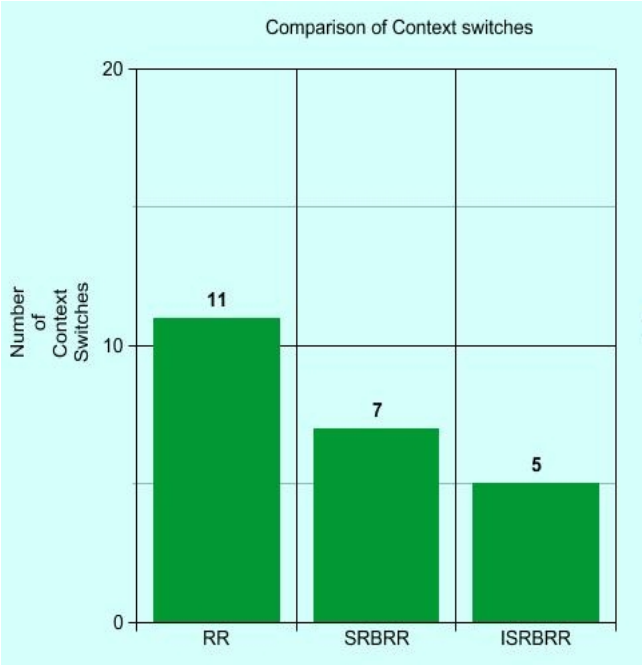


Figure: 1

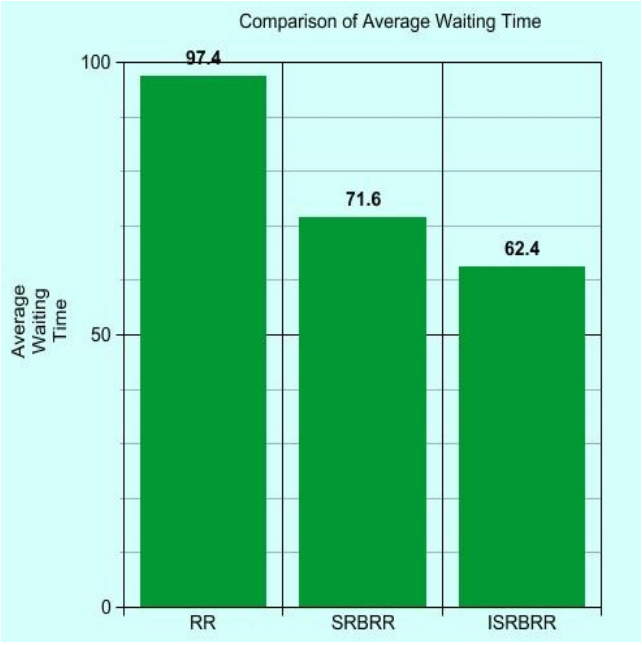


Figure: 2

Figure: 3

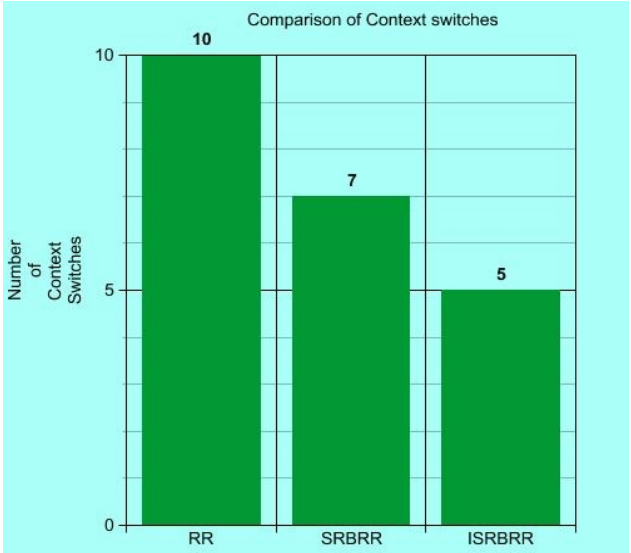


Figure: 4

Case II :

Let us assume five processes arriving at time = 0, with decreasing burst time (P1 = 86, P2 =53, P 3 = 32, P4= 21, p5= 9) as shown in TABLE

Table: 3

Process	Burst Time
P1	86
P2	53
P3	32
P4	21
P5	9

Now , TQ can be calculated as follows :
TQ = Ceil((Highest B.T + median)/ 2)
TQ = Ceil ((86+32)/2) = 59

P5	P4	P3	P2	P1	P1

0 9 30 62 121 180 207

Number of Context Switches = 5
Average Waiting Time = (0+9+30+62+121) / 5 = 44.4
Average Turnaround Time = (9+30+62+121+207) / 5 = 85.8
Table II: Comparison between RR, SRBRR and Proposed algorithm
(case – II)

Table: 4

Algorithm	Time Quantum	Avg.TAT	Avg.WT	CS
RR	25	150.8	110.5	10
SRBRR	32	89.8	49.6	7
ISRBRR	59	85.8	44.4	5

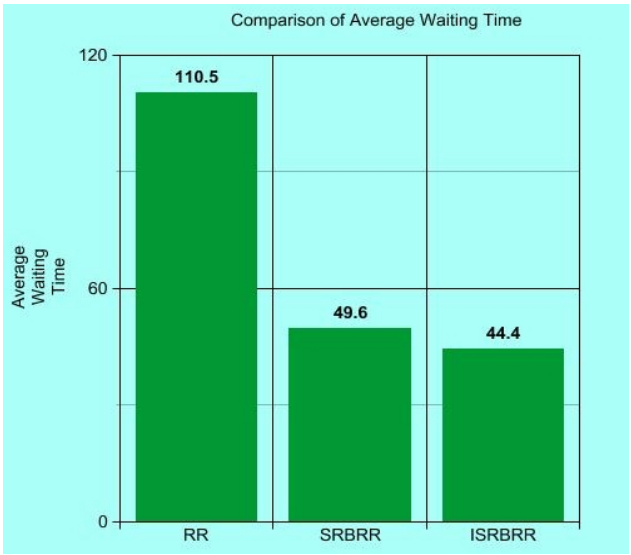


Figure: 5

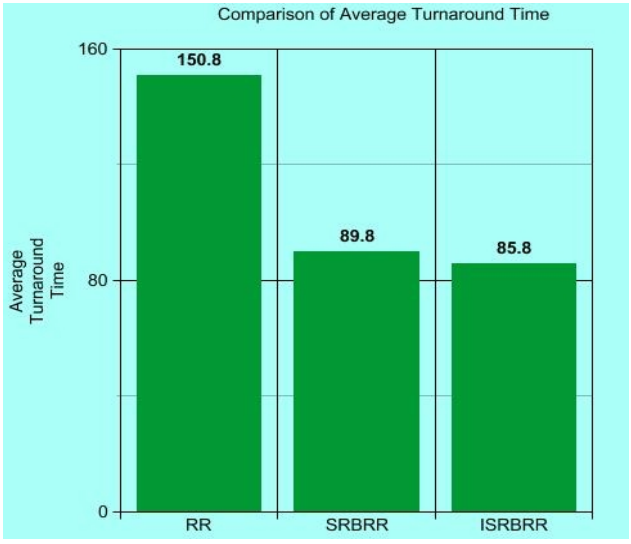


Figure: 6

Case-III:

Let us Assume five processes arriving at time = 0, with random burst time (P1 = 54, P2 = 99, P 3 = 5, P 4 = 27, p5= 32) as shown in TABLE

Table: 5

Process	Burst Time
P1	54
P2	99
P3	5
P4	27
P5	32

Now, TQ can be calculated as follows:
TQ = Ceil((Highest B.T + median)/ 2)
TQ = Ceil((99+32)/2) = 66

P3	P4	P5	P1	P2	P2

0 5 32 64 118 184 217

Number of Context Switches = 5
Average Waiting Time = (0+5+32+64+118) / 5 = 43.8
Average Turnaround Time = (5+32+64+118+217) / 5
= 87.2

Table III: Comparison between RR, SRBRR and Proposed algorithm
(case – III)

Table: 6

Algorithm	Time Quantum	Avg.TAT	Avg.WT	CS
RR	**25**	**152.2**	**108.8**	**11**
SRBRR	**32**	**93.6**	**50.2**	**7**
ISRBRR	**66**	**87.8**	**43.8**	**5**

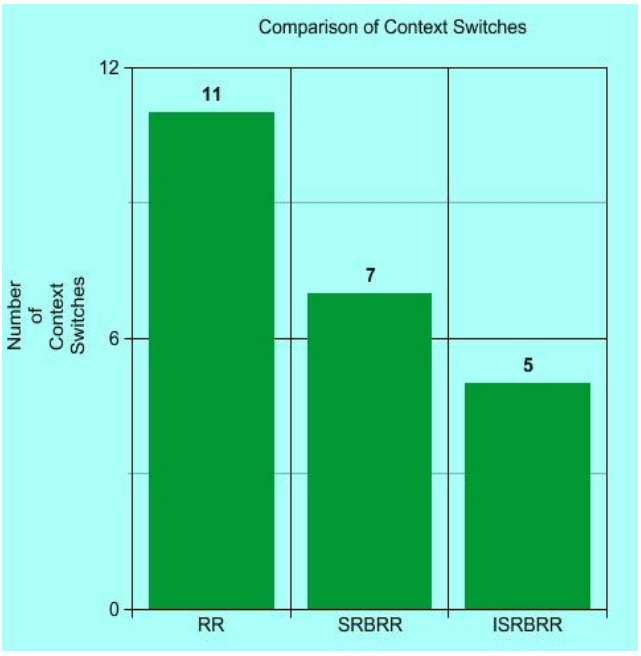


Figure: 7

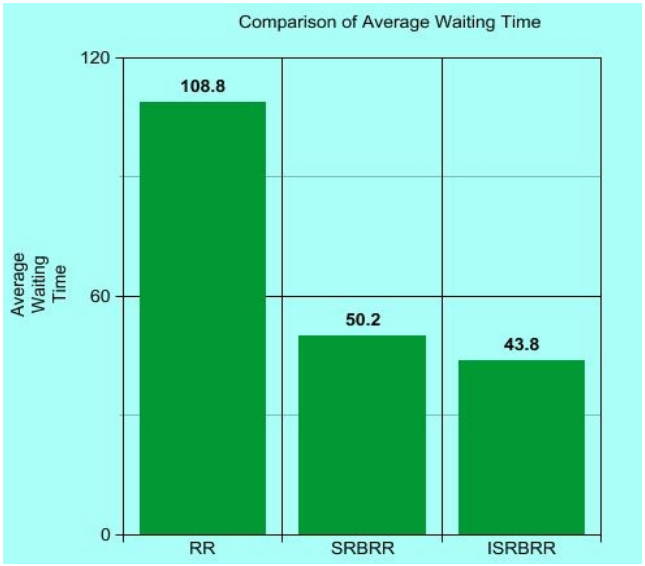


Figure: 8

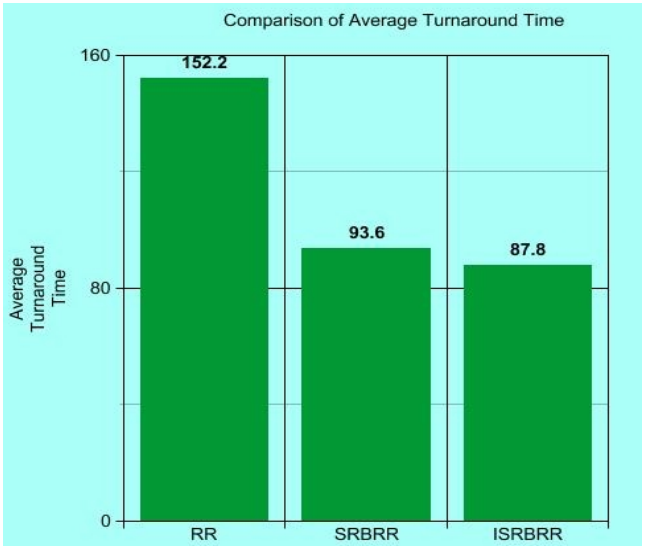


Figure: 9

C. Implementation:

The algorithm is implemented using C language and its code is as follows:

Source Code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int st[10];
int get_tq(int b[],int s)
{
int i,j,tmp,hbt,median;
float k,l,m;
for(i=0;i<s;i++)
{
for(j=i+1;j<s;j++)
{
 if (b[i]>b[j])
 {
  tmp=b[i];
  b[i]=b[j];
  b[j]=tmp;
   }
}
```

```
}
hbt=b[i-1];
median=b[i/2];
for(i=0;i<s;i++)
st[i]=b[i];
l=(float)hbt;
m=(float)median;
k=ceil((l+m)/2);
return(ceil(k));
}
void main()
{
int bt[10],wt[10],tat[10],n,tq;
int i,count=0,swt=0,stat=0,temp,sq=0;
float awt=0.0,atat=0.0;
clrscr();
printf("Enter number of processes:");
scanf("%d",&n);
printf("Enter burst time for sequences:");
for(i=0;i<n;i++)
{
scanf("%d",&bt[i]);
st[i]=bt[i];
}
tq=get_tq(st,n);
printf("\ntime quantum is computed by ceil((highestbt+Median)/2) = %d\n",tq);
while(1)
{
for(i=0,count=0;i<n;i++)
{
temp=tq;
if(st[i]==0)
{
count++;
continue;
}
if(st[i]>tq)
st[i]=st[i]-tq;
else
if(st[i]>=0)
{
temp=st[i];
st[i]=0;
}
sq=sq+temp;
tat[i]=sq;
}
if(n==count)
break;
}
for(i=0;i<n;i++)
{
wt[i]=tat[i]-bt[i];
swt=swt+wt[i];
stat=stat+tat[i];
}
awt=(float)swt/n;
atat=(float)stat/n;
//printf("Process_no\t Burst time\t Wait time\t Turn around time\t");
//for(i=0;i<n;i++)
//printf("%d\t %d\t %d\t %d\t",i+1,bt[i],wt[i],tat[i]);
printf("\nAvg wait time is %f\n Avg turn around time is %f\t",awt,atat);
getch();
}
```

OUTPUT

```
Enter number of processes:5
Enter burst time for sequences:13
35
46
63
97
time quantum is computed by ceil((highestbt+Median)/2) = 72
Avg waiting time is 62.400002
Avg turn around time is 113.199997
```

D. Simulation and Screen shots:

Turbo C++ is used in order to simulate the source code. Here are some screen shots of simulation process.

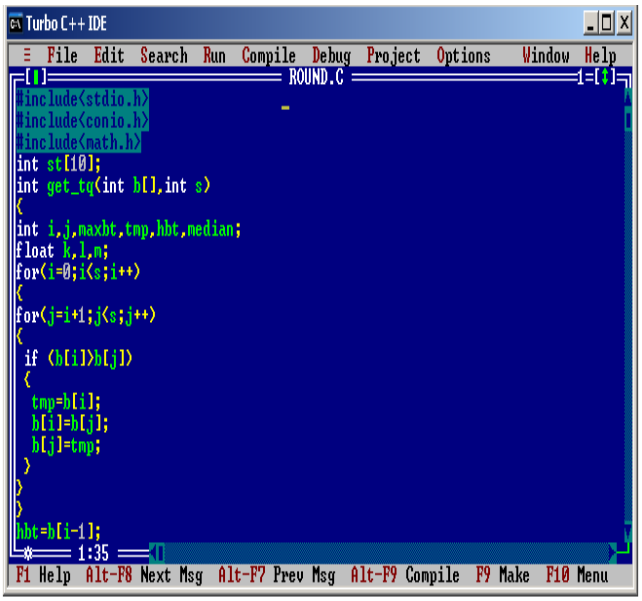


Figure: 10

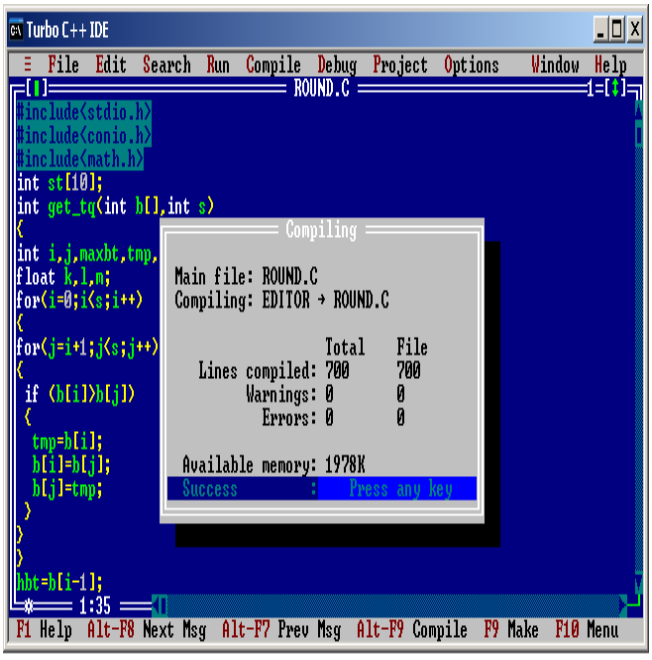


Figure: 11

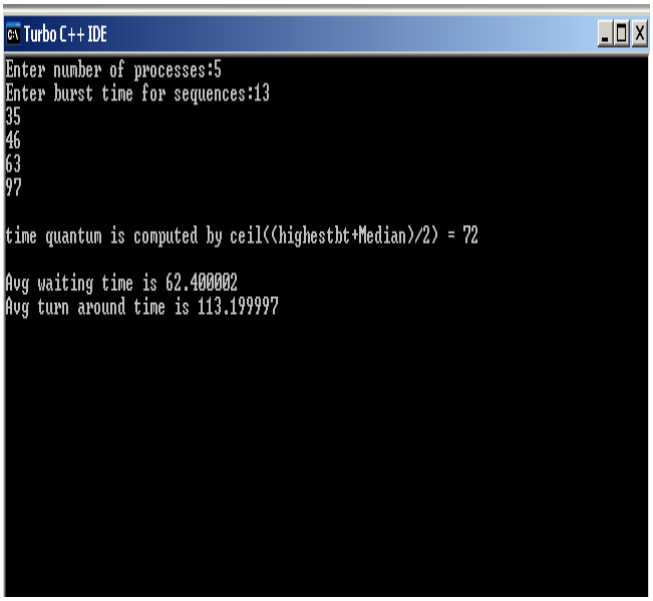


Figure: 12

VI. CONCLUSION AND FUTURE WORK

From the above comparisons I can conclude that the proposed algorithm is performing better than the static RR algorithm [1],[2],[5] and SRBRR algorithm [3] in terms of average waiting time, average turnaround time and number of context switches. In future work, processes at different arrival times can be considered for the proposed algorithm.

VII. REFERENCES

[1]. "Silberschatz, A., P.B. Galvin and G. Gagne, 2004" Operating Systems Concepts. 7th Edn., John Wiley and Sons, USA., ISBN: 13: 978-0471694663, pp: 944.

[2]. "Tanebaun, A.S., 2008" Modern Operating Systems. 3rd Edn., Prentice Hall, ISBN: 13: 9780136006633, pp:1104.

[3]. "Prof. Rakesh Mohanty, Prof. H. S. Behera, Khusbu Patwari, Manas Ranjan Das, Monisha Dash, Sudhashree" Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin(SRBRR) Scheduling Algorithm, Am. J. Applied Sci., 6 (10): 1831-1837, 2009.

[4]. "Yaashuwanth .C & R. Ramesh" Inteligent time slice for round robin in real time operating system, IJRRAS 2 (2), February 2010.

[5]. William Stallings, "Operating Systems: Internals and Design Principles" 6th edition, Prentice Hall, ISBN-13:978-0136006329