PROJECT FINAL REPORT

COMPARATIVE ANALYSIS
AND
IMPROVED SCHEDULING ALGORITHM
(SD BASED HRRN)

Contents

1.ABSTRACT

Most of the existing CPU scheduling algorithms are not applicable in real time operating systems because of their large waiting time, large turnaround time, large response time, high context switch rates and less throughput. The primary aim of this paper is to come up with a method for round robin CPU scheduling which enhances the CPU performance in a real time operating system. The proposed algorithm for real time systems is a hybrid of Standard deviation and HRRN scheduling algorithms.Itcombines the merits of standard devation, like elimination of starvation, and priority scheduling. The proposed algorithm also deals with aging by assigning variable priorities to the processes. Thus the algorithm corrects all the limitations of CPU scheduling algorithm. We will also present a comparative analysis of our new algorithm with the existing algorithms on the basis of average turnaround time, average waiting time, varying time quantum and number of contextswitches.

and we also compare our algorithms with other scheduling algorithms.

Key words: CPU Scheduling, Priority Scheduling.

2.INTRODUCTION

In computer science, scheduling is the process by which processes are given access to system resources like processor cycles, communications bandwidth. Hence, there arises the need for a scheduling algorithm for the computer systems to perform multitasking and multiplexing.

Scheduling is a fundamental operating system function that determines which process run, when there are multiple run able processes. CPU scheduling is important because it impacts resource utilization and other performance parameters. There exists a number of CPU scheduling algorithms like First Come First Serve, Shortest Job First Scheduling, Round Robin scheduling, Priority Scheduling etc, but due to a number of disadvantages these are rarely used in real time operating systems except Round Robin scheduling.

a number of assumptions are considered in CPU scheduling which are as follows:

- Job pool consists of run able processes waiting for theCPU.
- all processes are independent and compete forresources.
- The job of the scheduler is to distribute the limited resources of CPU to the different processes fairly and in a way that optimizes some performance criteria.

The scheduler is the component of the kernel that selects which process to run next. Operating systemsmayfeature up to three distinct types of schedulers, a long-term scheduler, a mid-term or medium-term scheduler and a short-term scheduler. The long-term scheduler or job scheduler selects processes from the job pool and loads them into memory for execution. The short-term scheduler, or CPU scheduler selects from among the processes that are ready to execute, and allocates CPU to one of them. The medium-term scheduler removes processes from memory and reduces the degree of multiprogramming results in the scheme of swapping. Swapping is the scheme which is performed by dispatcher which is the module that gives control of the CPU to the process selected by the short-termscheduler.

The CPU scheduling also plays an important role in the real time operating system which always has a time constraint on computations. a real time system is the one whose applications are mission-critical, where real-time tasks should be scheduled to be completed before their deadline. Most real-time systems control unpredictable environments and may need operating systems that can handle unknown and changing tasks. So, not only a dynamic task scheduling is required, but both system hardware and software must adapt to unforeseen configurations.

There are two main types of real-time systems: Hard Real-Time System, Firm or Soft Real-Time System. In Hard Real-Time System, it requires that fixed deadlines must be met otherwise disastrous situation may arise whereas in Soft Real-Time System, missing an occasional deadline is undesirable, but nevertheless tolerable. System in which performance

is degraded but not destroyed by failure to meet response time constraints is called soft real time systems.

In real time systemseach task should be invoked after the ready time and must be completed before its deadline, an attempt has been made to satisfy these constraints. Simple round robin architecture is not suitable to implement in Soft real time due to more number of context switches, longer waiting and response times. This in turn leads to low throughput in the system. If a real-time process having relatively larger CPU burst it will leads to the problem of starvation. Priority schedulingmaybe a better option for real-time scheduling but it will face the similar problem i.e. low priority processes will alwaysstarve.

2.2 SCHEDULINGOBJECTIVES

a system architect must consider various elements in constructing a scheduling algorithm, for example, kind of frameworks utilized and client necessities. Relying upon the system, the client and designer may anticipate that the schedulerswill:

- Maximize throughput: a scheduling algorithm ought to be equipped for adjusting the greatest number of jobs per unit oftime.
- avoid inconclusive blocking or starvation: a job ought not sit tight for unbounded time before or while processservice.
- Minimize overhead: Overhead causes wastage of assets. Be that as it may, when we utilize system resources viably, then general system execution enhances extraordinarily.
- Enforcement of priorities: if system allocates priorities to forms, the booking system ought to support the higher-needforms.
- attain balance between response, utilization: the scheduling algorithm must keep resourcesoccupied
- Support jobs which show desirablebehaviour.
- Corrupt gracefully under hugeload.

a system might finish these objectives in many ways. The scheduler can stall indefinite blocking of jobs via aging. Scheduler increments throughput by favouring processes whose requests can be satisfied quickly, or whose completion cause other processes to run.

2.3 SCHEDULINGCRITERIa

CPU scheduling algorithms maybe such which have different properties, and the choice of one particular algorithm might favour one category of processes over other. Choosing an algorithm for a particular situation, we must cater to properties of various algorithms. The scheduling criteria include the following:

- Context Switch: a context switch is process of storing and restoring context (state) of a pre-empted process, so that execution may be resumed from same point at a later stage. Context switching involves a lot of computation, lead to wastage of time and memory, which in turn increases the overhead of scheduler, hence operating system’s design, is to optimize only theseswitches.

- Throughput: it is the number of processes completed per unit time.Itis slow in round robin scheduling implementation. Context switching and throughput are inversely proportional.
- CPU Utilization: Tells the business of the CPU. One needs to maximize CPU utilization.
- Turnaround Time: Total time spent to complete the job. The time interval from the time of submission of a job to its completion is the turnaround time. Total turnaround time is the sum of the times spent waiting to get into memory, waiting time in the ready queue, execution time on the CPU and doingI/O.
- Waiting Time: Time a job has been stalled in ready queue. The CPU scheduling algorithm does not affect the amount of time during which a process executes or does input-output; it affects only the amount of time that a process spends waiting in readyqueue.
- Response Time: in real time systems, turnaround time may not be best measure. Often, a job can provide with some output fairly early and continue computing new results while previous results are being produced to the user. Thus, it is the time from the submission of a request until the first response is produced that means time when the task is submitted until the first response is received. Thus, the response time should beless.

Hence a good scheduling algorithm for real time and time sharing system must have: -

- less contextswitches.
- high CPU utilization.
- highthroughput.
- Less turnaroundtime.
- Less waitingtime.

3. ROUND ROBIN SCHEDULINGaLGORITHM

The round robin algorithm was designed mainly for time-shared systems not for real time systems. Time slice or time quantum is defined in case of RR algorithm, which refers to duration for which the process is allocated to the CPU and executed.

The processes which have to be executed are kept in a circular queue which has a head and a tail. The CPU scheduler will go around the queue, allocating the CPU to each process for a time interval of one quantum but the problem is that all the processes are arranged in FCFS (First Come First Serve)manner.

arriving processes are then added to the tail of the queue.

The CPU scheduler will then select the Process Control Block from the head of the ready queue. This is a disadvantage in RR algorithm since all processes are basically given the

same priority. Round robin also favours the process with short CPU burst and penalizes long ones.

Disadvantages:

The disadvantages of round robin CPU scheduling algorithm which affects execution process time shared system are asfollows-

- Larger waiting, response time and high rates of contextswitching.
  Since Real-time programs must guarantee response within specified time constraints therefore larger waiting, response time and high rates of context switching affect the system's performance and delay the results.
- Lowthroughput

Throughput is defined as number of process completed per time unit. If round robin is implemented in soft real time systems throughput will be low which leads to severe degradation of system performance because of high context switching. If the number of context switches is low then the throughput will be high. Context switch and throughput are inversely proportional to each other.

With these observations it is found that the existing simple round robin architecture is not suitable for real time.

4. PRIORITY SCHEDULINGaLGORITHM

The operating system assigns a fixed priority number to every process, and the scheduler then arranges the processes in the ready queue in order of their priority. These processes are then allocated to the CPU one by one. Lower priority processes get interrupted by incoming higher priorityprocesses.

Waiting time and response time in this algorithm depend on the priority of the processes. Higher priority processes have smaller waiting and response times. Deadlines can be easily met by giving higher priority to the earlier deadline processes.

Disadvantage:

Starvation of lower priority processes is possible if large number of higher priority processes keep arriving continuously.

Therefore a combination of the above two algorithms will be needed to make an algorithm fit for real time systems.

5. Shortest Job First(SJF)

Whichever process has the shortest burst time is scheduled

first. It works in two modes

1.Non-preemptive

2.Preemptive.

SJF Non-Preemptive

The process which has the shorter burst time is scheduled first. When the first process finishes its execution the next process with the shortest time is scheduled next

SJF Preemptive

In preemptive SJF the burst time of process are checked after every unit of time. after completing one unit check process having the shortest burst time will be scheduled next. It's also called as Shortest Remaining Time First.

6. Longest Job First(LJF)

Whichever process has the longest burst time is scheduled first. It works in two modes non- preemptive and preemptive.

LJF Non-Preemptive

The process which has the longest burst time is scheduled first. When the first process finishes its execution the process with the longest time is scheduled next.

LJF Preemptive

In preemptive LJF the burst time of process are checked after every unit of time. after completing one unit check process having the longest burst time will be scheduled next. It's also called as Longest Remaining Time First.

7. HRRN

Highest Response Ratio Next (HRRN)

This algorithm implements the "aging priority" scheme, in that as a process waits,

its priority is boosted until it eventually gets to run. The priority is calculated as follows:

$$\text{Priority} = (w + s) / s$$

Where :

w = time spent waiting for the

processor s = expected service

time

This policy is quite beneficial in that long processes will age, and thus

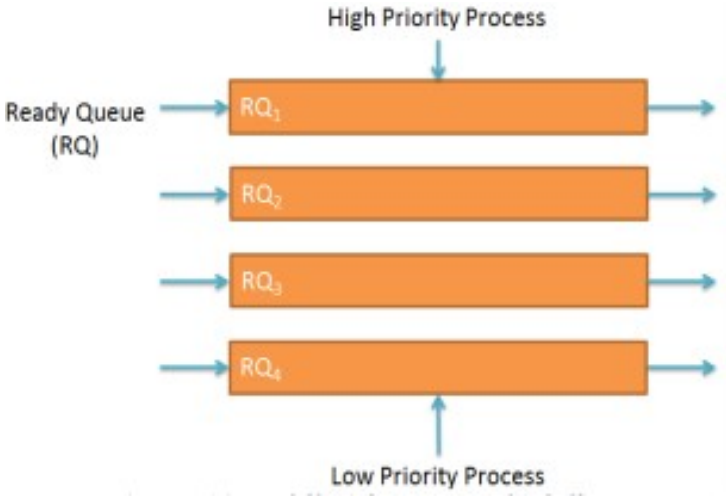
will eventually be assigned a higher-priority than the shorter jobs

(which already have a high-priority

because of the small denominator value).

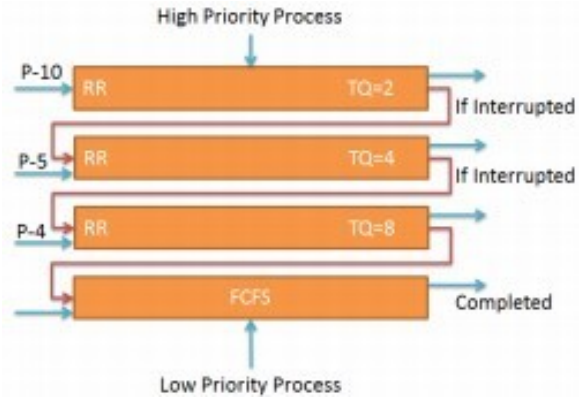
8. Multilevel QueueScheduling

In this process are partition into different queues and each has its own scheduling algorithm. Depending upon the priority of the process, in which ready queue the process has to be place, is decided. Generally, the high priority process is placed at the top level ready queue and low priority processes are placed in a bottom level ready queue. If this strategy is followed then the process are placed at bottom level ready queue will suffer from starvation.



9. Multilevel FeedbackQueue

This scheduling is same as for multilevel queue scheduling but the processes which have no completed its execution at the top level is interrupted and placed in the next level ready queue to remove starvation.



10. First Come First Serve(FCFS)

Whichever process comes first is scheduled first. The process to be scheduled first is decided by the arrival time of the processor.

Completion time is the time taken by the process to complete its execution. Turn around Time is the time difference between the completion time and arrival time. Waiting time is the difference between the turnaround time and the burst time. i.e. TaT=CT-aT and WT=TaT-BT.

11. Hybrid CPUScheduling:-

Hybrid CPU Scheduling algorithm is both preemptive and non-preemptive in nature. In this algorithm we find a factor known as Total Elapsed Time (TET) is calculated by the summation of burst time (B.T.) and arrival time(a.T.) i.e., TET = B.T. + a.T.. TET is assigned to each process and on the basis of TET process are sort in ascending order. Process having shortest TET is executed first and process with next shortest TET value is executed next. By considering the Burst Time (B.T.) the new algorithms acts as preemptive or non-preemptive. Proposed CPU scheduling algorithm decreases turnaround time, waiting time & response time and also increases CPU utilization and throughput. The working procedure of Hybrid CPU Scheduling of Non-Preemptive and Preemptive algorithm is as given below:

⎫Obtain the list of processes, their arrival time (a.T.) and burst time (B.T.).

⎫Find the Total Elapsed Time (TET) by summation of arrival time and burst time of processes.

⎫arrange the processes in ascending order based on TET.

⎫Take the processes for execution as follows Initially we assume that CPU arrival time is having some value (ZERO).

1.Pick lowestTET.

2.Compare Process arrival time with CPU arrival time is either equal orless.

3. If step 2 is not satisfied then, take next lowest TET and repeat step 2 until burst time of all processes becomezero.

4. If Total Elapsed Time (TET) of any two processes is equal and satisfied step 2 then execute process based on lowest processID.

12. SDScheduling

In SD scheduling, a queue is maintained in which processes are ordered on the basis of how close is the value of their burst time with the value of the standard deviation. Process whose burst time is close to the standard deviation value will be executed first and so on.

Standard Deviation is given by:

$$SD = \sqrt{\frac{\sum(X - \bar{X})^2}{N}}$$

Where:

$\bar{X}$is the mean value.

X is the burst time of each process. N is the number of the processes

13. JOB MIXScheduling:-

In Job Mix scheduling, a separate queue is maintained from the queue in which processes are kept in the order of lesser burst time first and then the higher burst time . all the processes are ordered in the same way and then the execution takes place. This method helps to eliminate the problem of starvation for the longer jobs and it provides better average waiting time which results in better performance of CPU. It is simple and easy to implement but there is increase in overhead in maintaining separatequeue.

14. OUR PROPOSEDALGORITHM:

The Highest Response Ratio Next (HRRN) was proposed to minimize the average value of the normalized turnaround time over all processes. For each process in the process pool, the response ratio, R is computed as:

$$R = (w + s)/s$$

Where w is the waiting time of the process in the ready queue and s is the expected burst time. Whenever the current process completes, the process with the greatest response ratio will be scheduled to run.

The response ratio, R is considered as the internal priority of a process and SD we taken here as Standard Deviation priority. The hybrid priority of each process is obtainedbygiving equal weight to both its sd priority and internal priority. The hybrid priority, Hp, of each process is computed as follows:

$$Hp= 0.5 * R + 0.5 * SD$$

Where R is the internal priority of each process and E is the external priority. Processes with lowest Hp are executed first

14.1 ALGORITHM

1. Begin

2. Initialize average waiting time, average Turn around time and average remaining time as zero

3. Processes arrive at the readyqueue.

4. Processes are sorted and assigned priority with the help of standarddeviation

5. Response ratio =(wt+bt)/wt.

6. Hybrid priority = 0.5 * Response ratio + 0.5 * SDpriority

7. If any two processes are with lowest Hybridpriority?

    If yes, did processes with same lowest hybrid priority arrives at same time?

        If yes Pi= any process with lowest Hybrid priority

        Else Pi= process with earliest arrival time among the processes

    Else Pi = process with lowest hybrid priority

8. Pi executes according to the bursttime

9. If burst time = 0 then process leaves ready queue calculate WT and TAT of Pi

   Go to 9

  Else go to 4

10. If ready queue is null, then calculate AWT, ART, ATAT of all the

    processes go to10

  Else go to 4

11. STOP

14.2 CODE

```
// CPP program for Highest Response Ratio Next
(HRRN) Scheduling #include <bits/stdc++.h>
using namespace
std; struct
process {
  char name;
  int at, bt, pt, ct,
  wt, tt; int
  completed;
  float ntt;
} p[10];
```

```
int n;

// Sorting Processes by
arrival Time void
sortByarrival()
{
  struct process
  temp; int i, j;

  // Selection Sort
  applied for (i = 0; i
  < n - 1; i++) {
    for (j = i + 1; j < n; j++) {

      // Check for lesser
      arrival time if (p[i].at >
      p[j].at) {

        // Swap earlier process to
        front temp =p[i];
        p[i] = p[j];
        p[j]
        =temp;
      }
    }
  }
}

int main()
{
  int i, j, t, sum_bt
  = 0; char c;
  float avgwt = 0,
  avgtt = 0; n = 6;
```

```
// predefined arrival
times int arriv[] = { 0,
0, 0, 0, 0, 0 };

// predefined burst times
int burst[] = { 3, 9, 6, 14, 10, 7 };

// predefined sd priority
int prior[] = { 1, 4, 2, 6, 5, 3 };

// Initializing the structure
variables for (i = 0, c = 'a'; i
< n; i++, c++) {
   p[i].name =
   c; p[i].at
   =arriv[i];
   p[i].bt =burst[i];
   p[i].pt =prior[i];



   // Variable for Completion status
   // Pending = 0
   // Completed = 1
   p[i].completed =0;

   // Variable for sum of all Burst
   Times sum_bt +=p[i].bt;
}

// Sorting the structure by
arrival times sortByarrival();
cout << "Name " << " arrival Time " << " Burst Time " << " Waiting Time "
<< " Turnaround Time " << "
Normalized TT" ; for (t = p[0].at; t <
sum_bt;) {
```

```
// Set higher limit to
response ratio float hrr =
9999;

// Response Ratio
Variable float temp;

// Variable to store next processs
selected int loc;
for (i = 0; i < n; i++) {

  // Checking if process has arrived and is
  Incomplete if (p[i].at <= t &&
  p[i].completed != 1) {

    // Calculating Response Ratio
    temp = (p[i].bt + (t -
    p[i].at)) / p[i].bt; temp =
    0.5*temp + 0.5*(p[i].pt);



    // Checking for Lowest
    Response Ratio if (hrr > temp)
    {

      // Storing Response
      Ratio hrr = temp;

      // Storing
      Location loc =
      i;
    }
  }
}
```

```
  // Updating time
  value t +=
  p[loc].bt;

  // Calculation of waiting
  time p[loc].wt = t -
  p[loc].at - p[loc].bt;

  // Calculation of Turn
  around Time p[loc].tt = t -
  p[loc].at;

  // Sum Turn around Time for
  average avgtt += p[loc].tt;

  // Calculation of Normalized Turn
  around Time p[loc].ntt =
  ((float)p[loc].tt / p[loc].bt);

  // Updating Completion
  Status p[loc].completed
  = 1;

  // Sum Waiting Time for
  average avgwt +=
  p[loc].wt;
  cout<< "\n" << p[loc].name <<"\t"
  << p[loc].at; cout << "\t\t" <<
  p[loc].bt <<"\t\t"<< p[loc].wt; cout
  <<"\t\t"<< p[loc].tt <<"\t\t"<<
  p[loc].ntt;
}
cout << "\naverage waiting time: " <<
avgwt / n << endl; cout <<"average Turn
around time:"<< avgtt / n;
```

```
}
```

15. ComparisonTable

S. No.	algorithm	Implementation Complexity	Preemption	Prior Knowledge of priorities	Starvation	Performance
1	**First Come First Serve**	Easy (FIFO Queue)	No	No	No	large average waiting time
2	**Shortest** JobFirst	Easy to implementin Batchsystem	No	No	Yes	Minimum average waiting time
3	**Shortest Remaining Time First**	Impossible to implement in interactive systems	Yes	No	Yes	Short jobs are given Preference
4	**Round Robin**	Easy	No	No	No	Fixed time to each process
5	**Longest Job First**	Easy	No	No	Yes	Large average turnaround time
6	**Longest Remaining Time First**	Impossible to implement in interactive or real time systems	Yes	No	Yes	Longer job are given preference
7	**Priority NonPreemptive**	Mildly Complex	No	Yes	Yes	Best for batch systems
8	**Priority Preemptive**	Complex	Yes	Yes	Yes	Good but problem of starvation

9	**MultiLevel Queue**	Complex	No	Yes	Yes	Good but process suffer from starvation
10	**MultiLevel Feedback Queue**	More Complex	No	Yes	No	Starvation problem is removed
11	**Hybridcpu scheduling**	Mildly complex	No	No	No	Best for batch systems
12	**Standard deviation**	Easy	No	No	Yes	Suffer from Starvation
13.	**Job Mix**	Mildly complex	No	No	No	Increase in over head
14.	**HRRN**	Easy	No	Yes	Yes	Increase in Throughput
15.	**SD based HRRN**	Easy	No	Yes	Yes	Will consider both waiting time and priority.

16. RESULTS ANDOBSERVATIONS:

16.1 Comparison

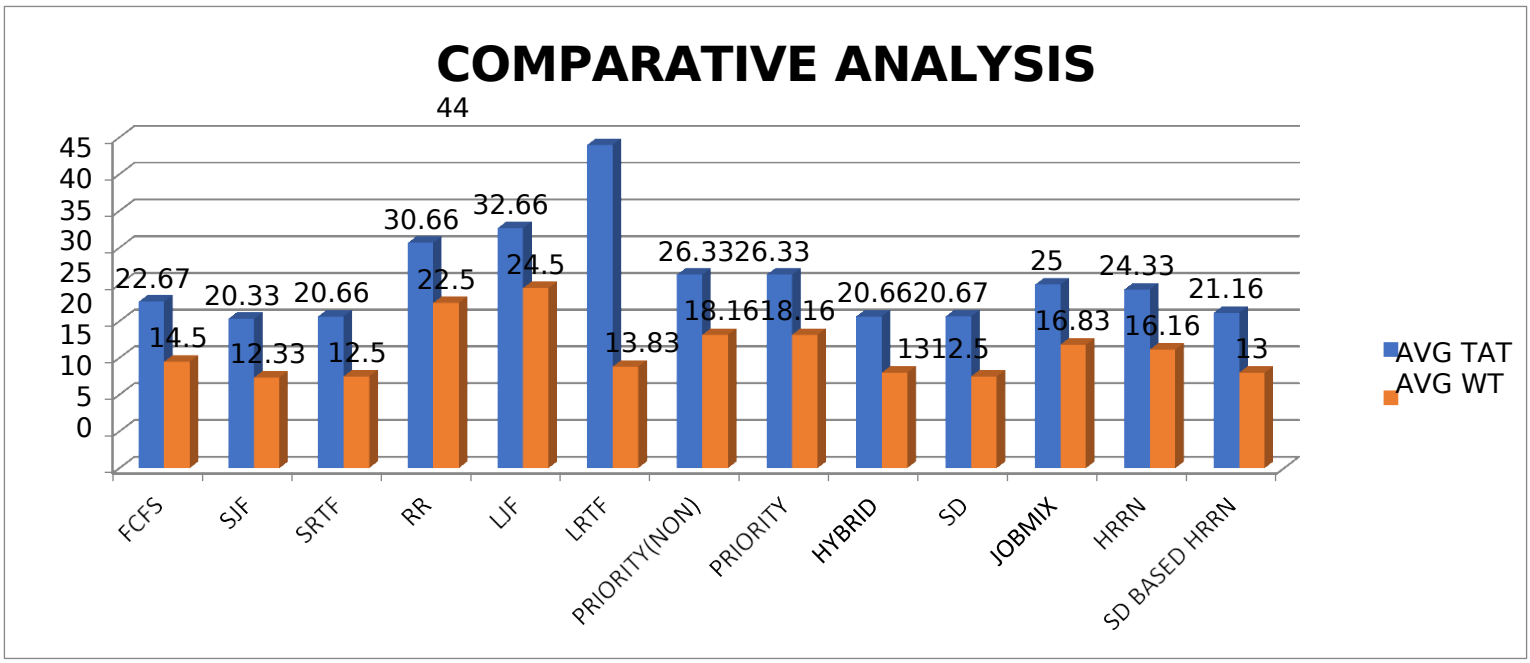
1. With arrivalTime:-

Process Number	arrival Time	Burst time
1	1	3
2	3	9
3	5	6
4	2	14
5	4	10
6	6	7

S.No	algorithm	average TaT	average Wt
1	**First Come First Serve**	**22.67**	**14.5**
2	**Shortest Job**	**20.33**	**12.33**

	First		

3	Shortest Remaining Time First	20.66	12.50
4	Round Robin	30.66	22.50
5	LongestJobFirst	32.66	24.5
6	Longest Remaining Time First	44	13.8333
7	Priority NonPreemptive	26.33	18.16
8	Priority Preemptive	26.33	18.16
9	Hybrid cpu scheduling	20.66	13
10	Standard deviation	20.67	12.5
11	Job Mix	25	16.833
12.	HRRN	24.33	16.16
13.	SD BaSED HRRN	21.16	13



Input/Output:-

```
Name  Arrival Time     Burst Time      Waiting Time   TurnAround Time    Normalized TT
A       1               3               0              3               1
B       3               9               1              10              1.11111
C       5               6               8              14              2.33333
F       6               7               13             20              2.85714
D       2               14              24             38              2.71429
E       4               10              36             46              4.6
Average waiting time: 13.6667
Average Turn Around time:21.8333
Process returned 0 (0x0)   execution time : 0.326 s
Press any key to continue.
```

2. Without arrival Time:-

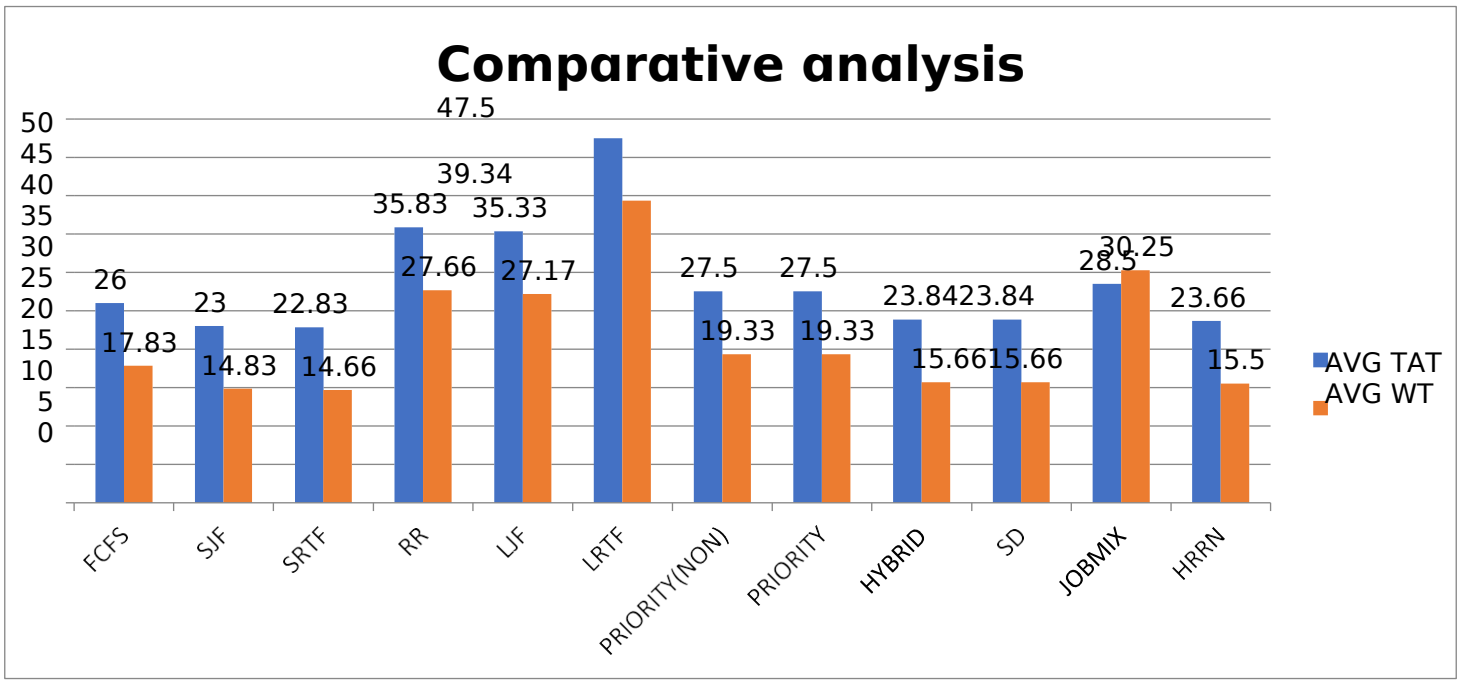
Process Number	Burst time
1	3
2	9
3	6
4	14
5	10
6	7

S.No	Algorithm	Average TAT	Average Wt
1	**First Come First Serve**	26	17.83
2	**Shortest** JobFirst	23	14.83
3	**Shortest Remaining Time First**	22.83	14.66
4	**Round Robin**	35.83	27.66
5	**Longest Job First**	35.33	27.17
6	**Longest Remaining Time First**	47.5	39.34
7	**Priority NonPreemptive**	27.5	19.33

8	**Priority Preemptive**		
9	**Hybrid cpu scheduling**	23.84	15.667
10	**Standard deviation**	23.84	15.667
11	**Job Mix**	28.5	30.25
12.	**HRRN**	23.66	15.5
13.	**SD BaSED HRRN**	23.5	15.33

Input/Output:-

```
Name  Arrival Time     Burst Time      Waiting Time    TurnAround Time     Normalized TT
A       0               3               0               3               1
C       0               6               3               9               1.5
F       0               7               9               16              2.28571
B       0               9               16              25              2.77778
D       0               14              25              39              2.78571
E       0               10              39              49              4.9
Average waiting time: 15.3333
Average Turn Around time:23.5
Process returned 0 (0x0)   execution time : 0.250 s
Press any key to continue.
```



17. CONCLUSION

From theabove comparisons, we observed that our new proposedalgorithm is performing better than the RR, FCFS ,LJF,LRTF, Priorityalgorithm proposed in paper in terms ofaverage waiting timeandaverage turnaround time thereby reducing the overhead, saving of memory spacesand solving the problem of starvation which was caused by Priorityalgorithm. When compared to the SJF, SRTF, HYBRIDand Standard Deviation have nearly same valuesas TaTand WT. But thesealgorithms have their own limitations like SJFand Standard Deviation has more starvation when compared to our new proposedalgorithm whereas hybrid scheduling has complexity in implementation.

Thus by consideringall the cases, the proposedalgorithm will meet the demands ofa SOFT REAL TIME SYSTEM.

18. REFERENCES:

[1] Silberschatz, a., Peterson, J. L., and Galvin, B.,Operating System Concepts, addison Wesley, 7th Edition,2006.

[2] E.O. Oyetunji, a. E. Oluleye," Performance assessment of Some CPU Scheduling algorithms", Research Journal of Information Technology, 1(1): pp 22-26,2009.

[3] William Stallings, Operating Systems Internal and Design Principles, 5th Edition ,2006.

[4] Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., and Weglarz, J.: Scheduling Computer and Manufacturing Processes, Berlin, Springer,(2001).

[5] Stallings, W.: Operating Systems Internals and Design Principles, 5th edition, Prentice Hall,(2004).

[6] Sumit Kumar Nager , Nasib Singh Gill, "Comparative Study of Various CPU Scheduling algorithm",(IJCSE) International Journal on Computer Science and EngineeringVolume 6, Issue 2, March - April2017.

[7] Dr. Kuldeep Singh Kaswan, amandeep, " a NEW TECHNIQUE FOR CPU SCHEDULING: STANDARD DEVIATION BASED",(IJCSE) International Journal on Computer Science and EngineeringVolume 6, Issue 8, August2017.

[8] Anil Kumar Gupta, " Hybrid CPU Scheduling algorithm "(IJCSE) International Journal on Computer Science and EngineeringVol. 6 (2) , 2015,1569-1572.

[9] Neetu Goel , Dr. R. B. Garg , " Performance analysis of CPU Scheduling algorithms with Novel OMDRRS algorithm " ,(IJCSE) International Journal on Computer Science and EngineeringVol. 7, No. 1,2016.

[10] Silberschatz, a. P.B. Galvin and G. Gagne (2012), Operating System Concepts, 8th edition, Wiley India