REPORT 2

Dr. Manish Kumar Bajpai

PDPM IIITDM Jabalpur

Ankit Kesarwani (2015032)

Vivek Shukla (2015281)

}

Implementation of Method 3

Problem Statement:

Minimizing the Response time thereby reducing the *Waiting time* for lowest priority processes for existing Priority Scheduling Algorithm.

Implementation:

```
#include<bits/stdc++.h>
using namespace std;
struct process{
	int pid;				// process id
	int priority;			// process priority
	double bt;				// process burst time
	double wt;				// process waiting time
	double rt;				// process remaining time
	double tat;				// process turn around time
};
bool compare(struct process p1,struct process p2){
	return p1.priority>p2.priority?0:1;							//
sorting according to priority
}
int main(){
```

```
	int n;
	cout<<"enter the number of processes"<<endl;
	cin>>n;
	struct process p[n];
	cout<<"enter burst time and priority of each process 
respectively"<<endl;
	for(int i=0;i<n;i++){
		p[i].pid=i+1;
		cin>>p[i].bt>>p[i].priority;					// 
taking input process burst time and priority
		p[i].rt=p[i].bt;
	}
	sort(p,p+n,compare);
	double bt_min=INT_MAX,bt_max=INT_MIN;
	for(int i=0;i<n;i++){
		if(p[i].bt<bt_min){
			bt_min=p[i].bt;
		}
		if(p[i].bt>bt_max){
			bt_max=p[i].bt;
		}
	}
	queue<int> q;
	double time_slice=(bt_max+bt_min)/2;
	double time=0;
	for(int i=0;i<n;i++){
		if(time_slice>=p[i].rt){
```

```
                    time=time+p[i].rt;
                    p[i].rt=0;
        //first cycle
                    p[i].tat=time;
                    p[i].wt=time-p[i].bt;
                    q.push(p[i].pid);
                }
                else{
                    p[i].rt=p[i].rt-time_slice;
                    time=time+time_slice;
                    q.push(p[i].pid);
                }
        }
        int pid,tat,bt,priority,wt,rt;

        for(int i=0,j=n-1;i<n/2;i++,j--){
                pid=p[i].pid;
                tat=p[i].tat;
                bt=p[i].bt;
                priority=p[i].priority;
                wt=p[i].wt;
                rt=p[i].rt;
                p[i].pid=p[j].pid;                                  // swapping
    array
                p[i].tat=p[j].tat;
                p[i].bt=p[j].bt;
                p[i].priority=p[j].priority;
```

```
            p[i].wt=p[j].wt;
            p[i].rt=p[j].rt;
            p[j].pid=pid;
            p[j].tat=tat;
            p[j].bt=bt;
            p[j].priority=priority;
            p[j].wt=wt;
            p[j].rt=rt;

        }
        for(int i=0;i<n;i++){
            if(p[i].rt==0){
                continue;
            }
            else{
                time=time+p[i].rt;
                p[i].rt=0;                                          // cycle 2
                p[i].tat=time;
                p[i].wt=time-p[i].bt;
                q.push(p[i].pid);
            }
        }
        cout<<"pid   burstTime    priority    turnAroundTime    waitingTime
"<<endl;
        for(int i=0;i<n;i++){
            cout<<p[i].pid<<"      "<<p[i].bt<<"         "<<p[i].priority<<"
"<<p[i].tat<<"      "<<p[i].wt<<endl;
```

```
	}
	int sz=q.size();
	cout<<"printing gantt chart"<<endl;
	for(int i=0;i<sz;i++){
		cout<<q.front()<<" -> ";
		q.pop();
	}
	cout<<endl;

	return 0;
}
```

Case Study: -

Process No.	Priority	Arrival Time	Burst Time
P1	6	0	91
P2	4	0	67
P3	3	0	32
P4	7	0	28
P5	1	0	97

According to Priority Scheduling Algorithm:

Gantt Chart: -

P5	P3	P2	P1	P4

0 97 129 196 287 315

Process No.	Priority	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
P1	6	0	91	287	287	196	196
P2	4	0	67	196	196	129	129
P3	3	0	32	129	129	97	97
P4	7	0	28	315	315	287	287
P5	1	0	97	97	97	0	0

- Average Completion Time = 204.8
- Average Turn Around Time = 204.8
- Average Waiting Time = 141.8
- Average Response Time = 141.8

According to Proposed Priority Scheduling Algorithm:

Calculation of Time Slice:

Time Slice = (Burst time of shortest priority process + Burst time of largest priority process)/2

Time slice = (97 + 28) / 2 = 62.5

Gantt Chart: -

P5	P3	P2	P1	P4	P1	P2	P5

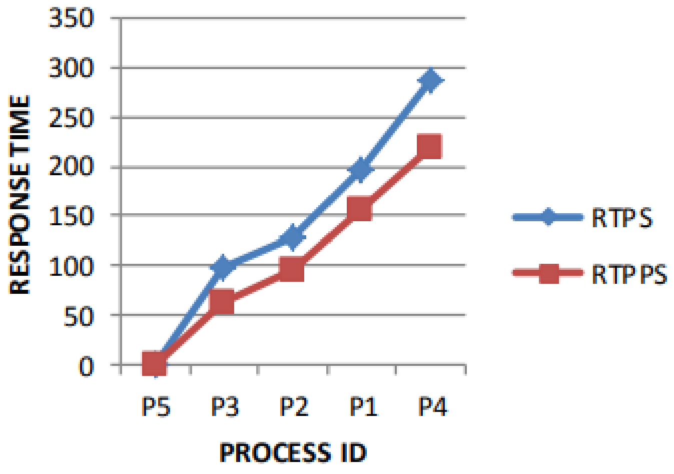
0 62.5 94.5 157 219.5 247.5 276 280.5 315

Process No.	Priority	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
P1	6	0	91	276	276	185	157
P2	4	0	67	280.5	280.5	213.5	94.5
P3	3	0	32	94.5	94.5	62.5	62.5
P4	7	0	28	247.5	247.5	219.5	219.5
P5	1	0	97	315	315	218	0

- Average Completion Time = 242.7
- Average Turn Around Time = 242.7
- Average Waiting Time = 179.7
- Average Response Time = 106.7

Analysis:

Algorithms	Average Turn Around Time	Average Waiting Time	Average Response Time	Context Switch
Priority Scheduling	204.8	141.8	141.8	4
Proposed Priority Scheduling	242.7	179.7	106.7	7



Comparison of Response Time of Priority Scheduling and Proposed Priority Scheduling Algorithm

Conclusion: -

- Response time for individual process is greatly reduced.
- Lowest priority processes are prevented from starvation.
- Waiting time for Lowest priority process is minimized.