OS-ASSIGNMENT-2

REPORT

ES19BTECH11003

Aim:The goal of this assignment is to implement a program to simulate the Rate-Monotonic & Earliest Deadline First (EDF) scheduling algorithms. Then compare the average waiting time and deadlines missed of both algorithms.

Low Level Design of a Program

We consider linked list implementation for implementation of priority queue

We use arrays in every function so it becomes easier if we define them globally.

So

```
int process_id[1000];//this is to store the process ID
int period[1000];//this is to store the period of each process
int execution_time[1000];//this is to store the execution time of each process
 int capacity[1000];//this is to store the k value
 int priority[1000];//this array is developed based upon the priority (i.e;period in RMS and deadline in EDF)
 int count_value[1000];//this stores that no of times that individual process executes
int waiting_time[1000];//this stores the waiting time of each individual process based upon index number
```

And we define struct node as

```
struct node
{
        struct node* next;
```

```
        int ID;
        int time;
        int priority;
}
```

Here we include int ID to process nodes which stores the actual process ID in priority queue and we include int priority which is the main part of implementation of priority queue as addition and deletion of nodes into priority queue will take place by comparing this priority of the nodes.and int time stores the execution time of the process in priority queue and keeps on decreasing as the process executes and when it becomes 0 it is considered that process has executed successfully.

Check_period() function to check the current running time is whether there are multiples of periods of processes. If yes then these new processes will be added into priority queue by using add_into_queue function.

add_into_queue() function initially creates a newnode and assigns all the values to the node based upon the process and starts searching for the position in the priority queue to where it should be added based upon the process priority and priorities of processes that are already present in the priority queue.

Execute() function makes the process present in the head of linked list(i.e; priority queue) to execute which means that it reduces the execution time of the process present in head by 1 and if the process completes then it changes the head.

waiting() function refreshes the waiting time of all the processes present in the linked list except the process present in head as that process executes correctly.

Challenges faced For Implementation of RMS Scheduling

While implementing algorithm for RMS scheduling the biggest challenge we face is that

adding a new process into the priority queue.

removing processes whose deadline is passed from the priority queue.

Calculation of waiting time

We can overcome the first 2 challenges by closely examining all the possibilities and then start implementing the priority queue.

And also while adding into the priority queue if 2 processes have equal priority then the process having least processID should be considered first.

And if a deadline is passed for a process we have to remove it from the priority queue considering the addition of a new process with the same process ID.

For calculation of waiting time we also have to consider a case when execution of head node is 1 unit left then along with changing of head we have to also add the waiting time of new head by 1 unit.

Challenges faced For Implementation of EDF Scheduling

Here while implementing the EDF scheduling algorithm the challenges faced are

Here priority of the nodes are nothing but their deadlines and so even if 2 nodes having same process ID differ in their node priority as their deadlines are different.

Removing of processes which passes their deadline is also the biggest challenge faced as here there are many number of cases which should be

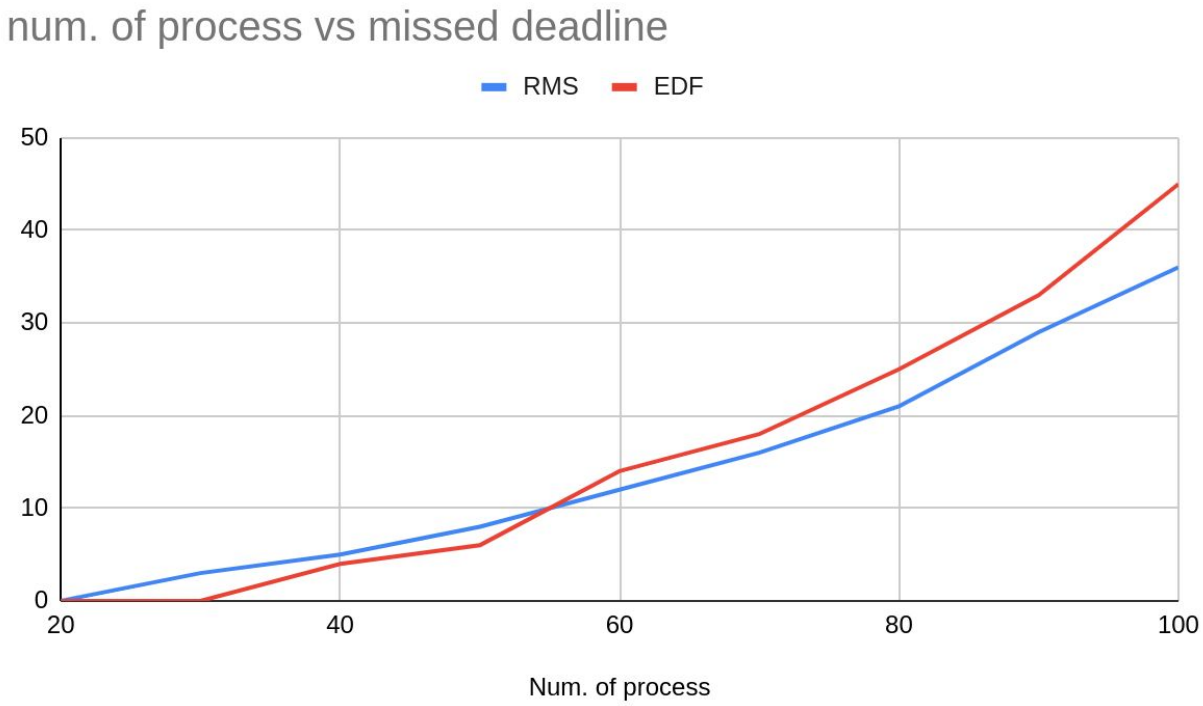
covered as priority is not fixed for a fixed process as the deadline for the same process keeps on changing.so we have to carefully consider all possible cases while adding a node into priority queue.and also while deleting the process from the node whose deadlines are passed we get to know that a process deadline is passed if we get a new process in the same priority queue having the same process ID of the old process whose deadline missed.

Also there is a bit difficulty in implementing a process exactly k times so to avoid this i have used count_value array which keeps the track of no of times each process executed based on the index number of the array.

GRAPHS

GRAPH-1

No of processes VS no of processes that missed their deadline in both RMS and EDF scheduling algorithms.



OBSERVATIONS FROM ABOVE GRAPH

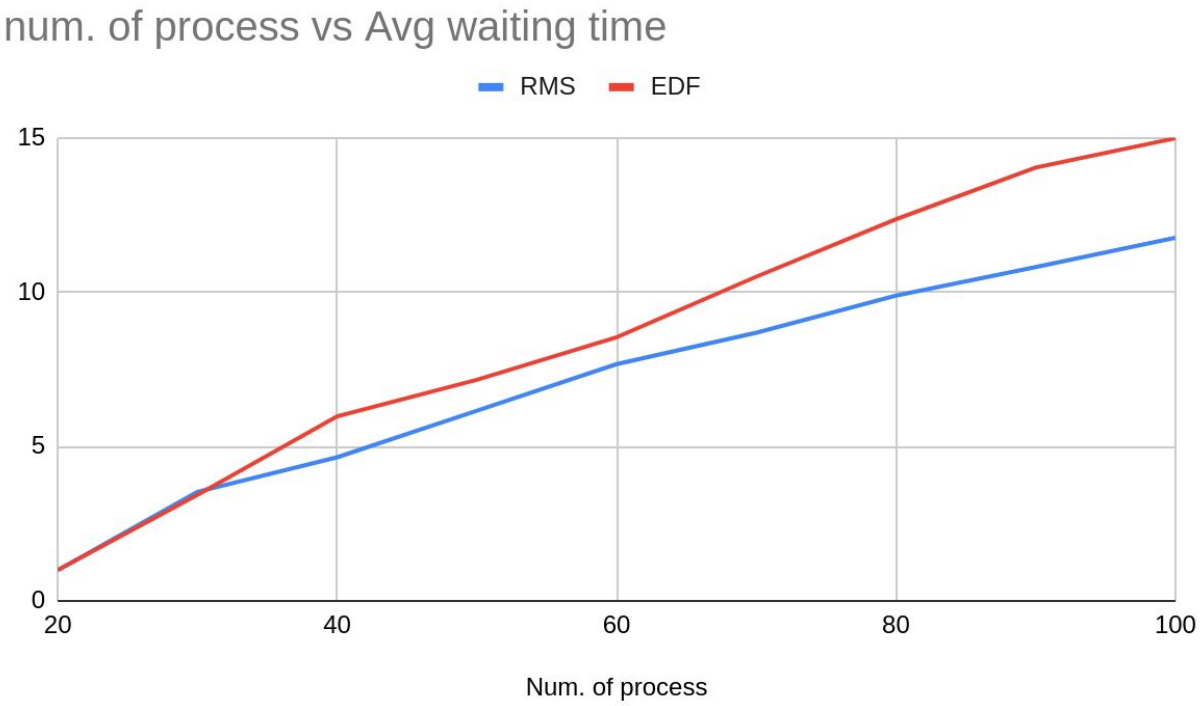
From the above graph we can clearly know that for fewer total processes RMS has more no of processes which are passing the deadline than EDF but as no of processes increases,processes which are missing their deadline will be more in EDF than RMS.

In general EDF should be far better than RMS because the priority in that case would be dynamic and we don't have the problem of starvation.

But according to our assumption we should not run any process after its deadline is over so there could be slight variation in the graph compared to the general.

GRAPH-2

No of processes VS average waiting time in both RMS and EDF scheduling algorithms.



OBSERVATIONS FROM ABOVE GRAPH

Initially for lower no of processes both EDF and RMS maintain same average waiting time of process but gradually as no of processes increases EDF has more waiting time as compared to RMS as

our assumption we should not run any process after its deadline is over so there could be slight variation in the graph compared to the general and also for processes whose deadline is issued slightly also we have to stop there and we have to add its period to its waiting time when it misses the deadline.

Hence in the graph we observe RMS having less avg waiting time when compared to EDF.

K also plays an important role in these average waiting times as a specific process has to execute exactly K times so processes

having far deadlines get least priority so waiting time slightly increases in case of EDF.

FINAL CONCLUSION

In discrete time scheduling and considering our assumptions RMS is slightly better in avg waiting time but in real time of scheduling EDF is far better when compared to RMS.