Operating Systems 2
Name :- Ajinkya Bokade
Roll No :- CS17BTECH11001

Design of Program

In this two programs, we have implemented rate montonic scheduling and earliest deadline first scheduling through discrete event simulation.
For a process, a class has been made which contains fields like the processes' cpu burst, its period, its deadline, its frequency (instances of that process), its remaining time for completion, its response time (time when it just starts to execute on cpu), its arrival time (in ready queue).
A vector of <double,string> (first parameter specifies the time and second the pid of process) is created for event queue named as event_timestamp.
It contains all the events when different instances of each process arrives into the system.
A vector of <double,process> (first parameter specifies period in case of RMS and deadline in case of EDF and second is the process) is created for ready queue. Hence after sorting the ready queue, the process with highest priority (least period in case of RMS and earliest deadline in case of EDF) would be in the beginning of ready queue.
A variable of type double current_time just jumps from event to event and considers following cases (events) and prints that events in log:-
1) A process starts its execution.
2) A process finishes its execution.
3) A process is preempted by another process of higher priority.
4) A process resumes its execution.
5) A process misses its deadline.

Rate Monotonic Scheduling

Rate monotonic scheduling is the policy where high priority is given to the process with least period. Hence, it is static scheduling since periods of process remain constant.

Earliest Deadline First :−
Earliest Deadline First is the policy where high priority is given to the process which have earliest deadline. In case of equal deadlines, in my implementation, priority is given as per first come first serve basis. Hence, it is dynamic scheduling since the deadlines of each instance of process is increased by the period of the process.

Comparison parameters

1) Average waiting time :−
It is defined as the time a process spends in the ready queue waiting and not executing.
Hence, average waiting time is defined as the total time spent by each instance of process in ready queue not executing divided by total instances of all processes.

2) Deadlines missed :−
A process is said to have missed its deadline if it cannot complete its execution upto its deadline.

Total number of processes is simple the summation of frequency of each process.
If a process has been succesfully completed, then we are incrementing the variable successes and if a process has missed its deadline, then we are incrementing the variable deadlines_missed.

Working of program

Implementation of Rate monotonic scheduling

We have created an event queue where the time (at which process arrives) and pid of process is stored for each instance of each process in sorted manner. So, we are running a loop while the ready is not empty or events (processes) are still remaining. Initially ready queue contains all processes since first instance of each process joined at time 0. At every iteration of loop, we are sorting the ready queue. As a result the process with highest

priority (least period) comes at the beginning. Then we are executing that process. Here, 4 cases as follows can occur:−
1) No event (process) comes in between the completion of process :−
In this case, we execute the process till its completion and remove it from ready queue and simply execute new process with highest priority from ready queue.

2) An event (process) occurs in between its execution but new process has low priority :−
In this case, we simply add the new process to the ready queue and allow the current process to run.

3) An event (process) occurs in between its execution but new process has high priority :−
In this case, we preempt the current process and adjust its remaining time for completion and allow the new process to run.

4) Current process can miss its deadline :−
Before adding any instance of process, we are checking in ready queue if any instance of same process is already present in queue. If its present, then we simply miss its deadline and remove it from ready queue and simple add new instance of it in ready queue.

Implementation of Earliest Deadline First

All things remain same as RMS except the ready queue, in this case, consists of vector of <double,process> where first parameter specifies the deadline of process. Hence after sorting teh ready queue, the first element contains the process which has earliest deadline.
Here, if a process cant complete its execution before its deadline, then it misses its deadline and remove it from ready queue.

Here, we have maintained an unordered map for sum of waiting time of all instances of process of particular pid. Hence for each instance of process, waiting time is calculated as the difference between the time it just starts to execute on CPU and the its

arrival time in ready queue which are stored as fields in class of process. Hence for each process, we are summing the waiting times of all its instances. Average waiting time is thus calculated by summing waiting times of each process (pid) and dividing it by total number of processes.

Difficulties faced while implementing

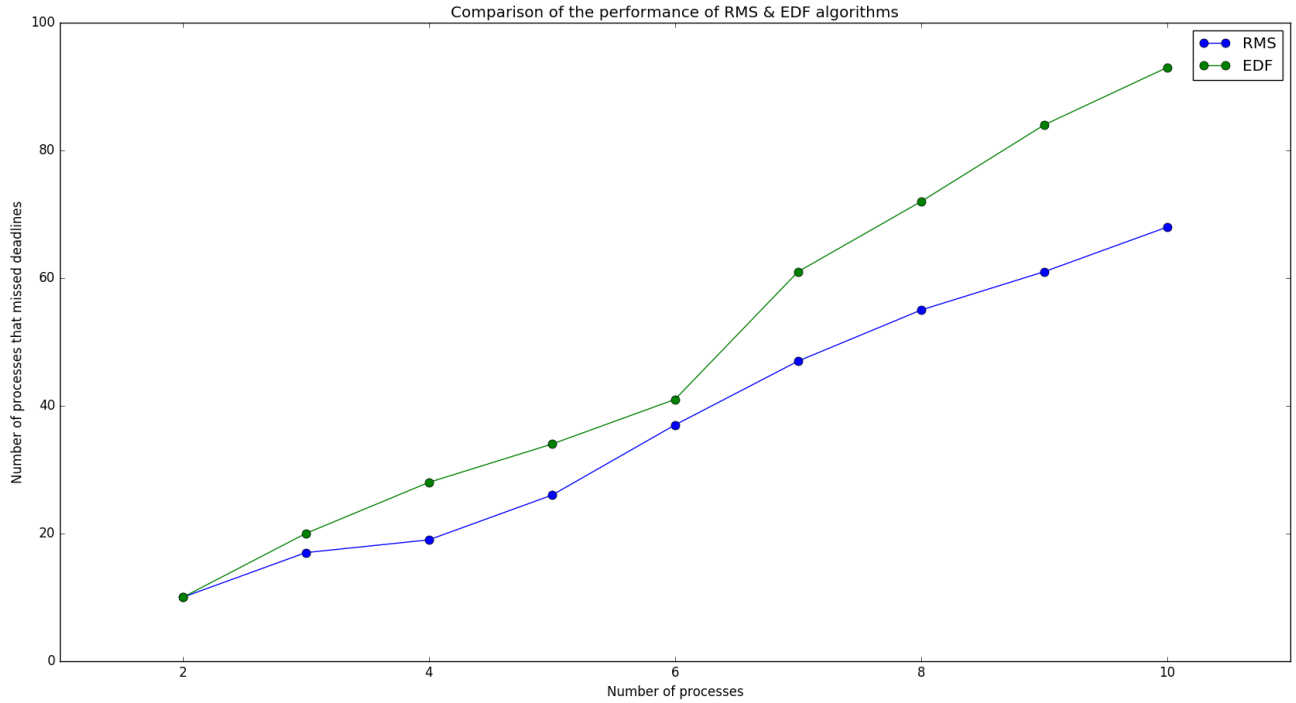
1) It was not quite logical to check if in between when current process is executing in CPU, any other process is missing its deadline since other processes are still in ready queue. So, its intutive to check deadline of the process which is executing in CPU.
2) RMS and EDF scheduling doesnt specify which process to prefer when the next incoming processes have equal period in case of RMS and equal next deadline. In my implementation, I have preferred the process which comes earlier and in case both processes have same arrival time also, then preference is given to the process which comes earlier in input.

Graphs

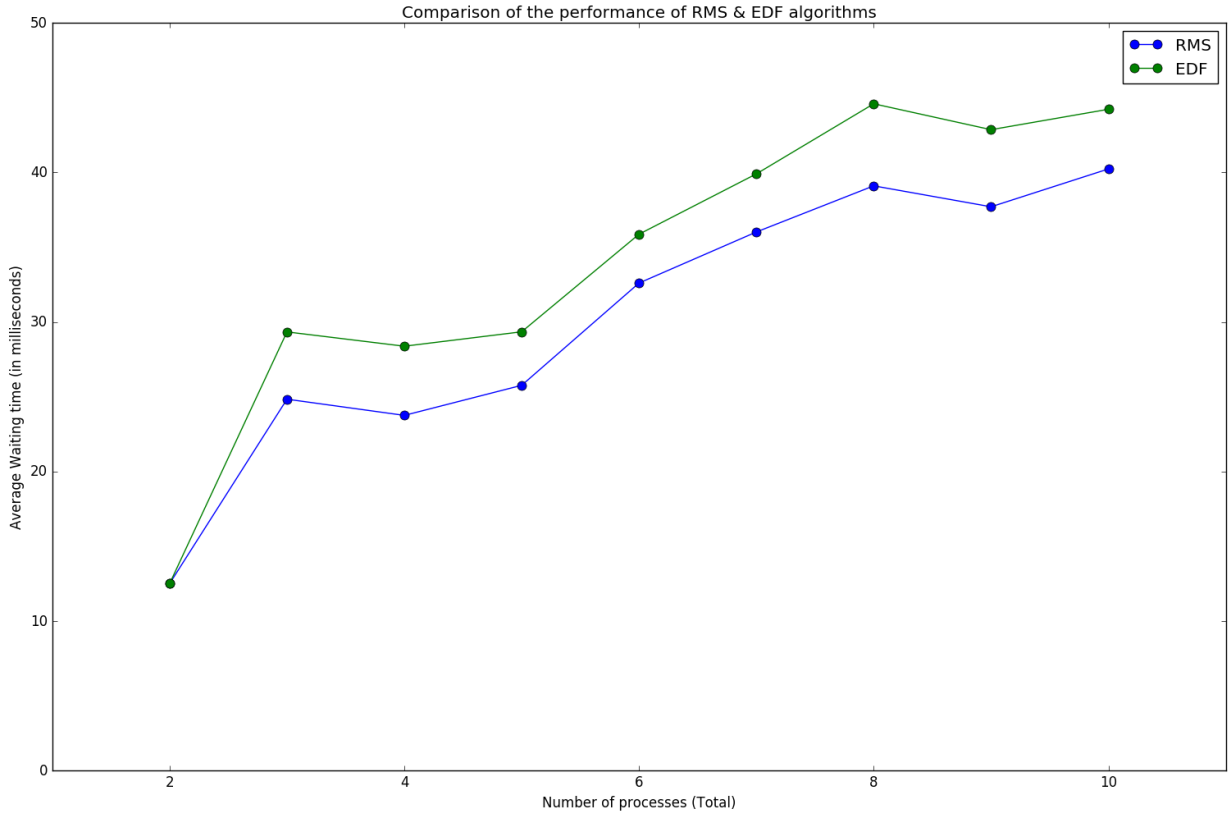
The following graphs are plotted for Rate Monotonic Scheduling and Earliest deadline first to compare the performance between the two.

Following graphs are plotted for data as given in 'inp−parameters.txt' for varying values of number of processes starting from 20 to 100 in increments of 10.

Graph 1 : Deadlines missed vs Number of processes



Graph 2 : Average waiting time vs Number of Processes



We observe from the graph that :−
1) For n=20 (Total Number of Processes), RMS and EDF both have 10 as missed deadlines and 12.5 ms as average waiting time. Thereafter, as n increases upto 100, number of deadline missed by processes for EDF are greater than that of RMS and also EDF has more average waiting time than RMS.
2) Rate Montonic Scheduling is better than Earliest Deadline First scheduling in terms of average waiting time as well as Number of Deadlines missed by processes.