Report

1. Design of the program

- The program consists of two classes namely process_type_info and process_instance. Class process_type_info consists of the general information about process and class process_instance consists of specific information about process, which nearly simulates PCB.
- There is operator overloading done to define the meaning of operator == in case of user-defined data type process_instance. The program also consists of structures GivePriority and GivePriorityProcessNotEntered. The structures are used to define priority for the processes in priority queues.
- The main function starts by taking input from “input.txt” file (if it exists) and saves the information in point, which is pointer to process_type_info. It actually has all the general information like PID, period, CPU burst time and no. of iterations. We are dynamically allocating memory to point.
- We then create two priority queues namely cpu and process_not_entered. Priority queue cpu sorts processes based on period, giving highest priority to one with least period for RMS. Priority queue cpu sorts processes based on period, giving highest priority to one with earliest deadline for EDF. It acts like process executor. Priority queue process_not_entered sorts processes based on entry time, giving highest priority to one with earliest entry time. It acts like process sender to cpu.
- We then push all the respective processes with their particular information into process_not_entered queue.
- We then create a while loop which runs until either cpu is not empty or all the iterations of processes are not done. We first check for completion of process if cpu is not empty. If any process is completed, then that process is removed from the cpu and completion of process is reported. We then have deadline check,

which checks if any process has missed the deadline. If yes, then it removes the process from the cpu and deadline miss is reported.

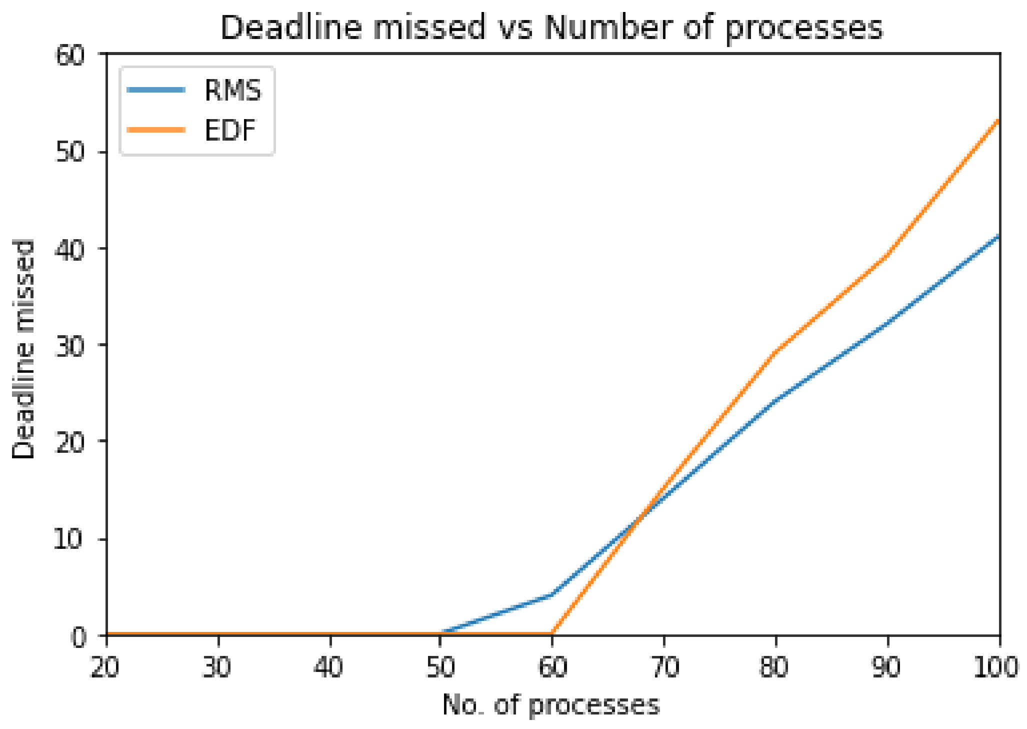
- The next part includes checking for new arrivals and preemption check due to new arrival. Preemption occurs if the new process which has arrived has higher priority over the process running in the cpu. If this is the situation, we preempt the process and report preemption. If there are new arrivals, then we push them into cpu.
- We then make processes to start or resume depending on whether they are preempted or not and report the same. At the end, we decrease the runtime left of the process at the top of the cpu and increment the time counter.
- We are printing the logistics of the scheduling to log file and statistics of the scheduling to stat file. Waiting time is calculated at the point of time where either process is completed or has missed its deadline. We calculate number of processes that have entered, are successful, have missed the deadline during particular checks in the while loop.
- Both RMS and EDF are similar but the change of priority makes them different.

2. Complications that arose

- Checking deadline miss for the processes that haven't even started execution in cpu but just present in cpu.
- Checking whether cpu is idle was easy but took little time to make the calculation easier.
- Priority queues don't allow to change the elements in them. This is because it stores the elements as const.

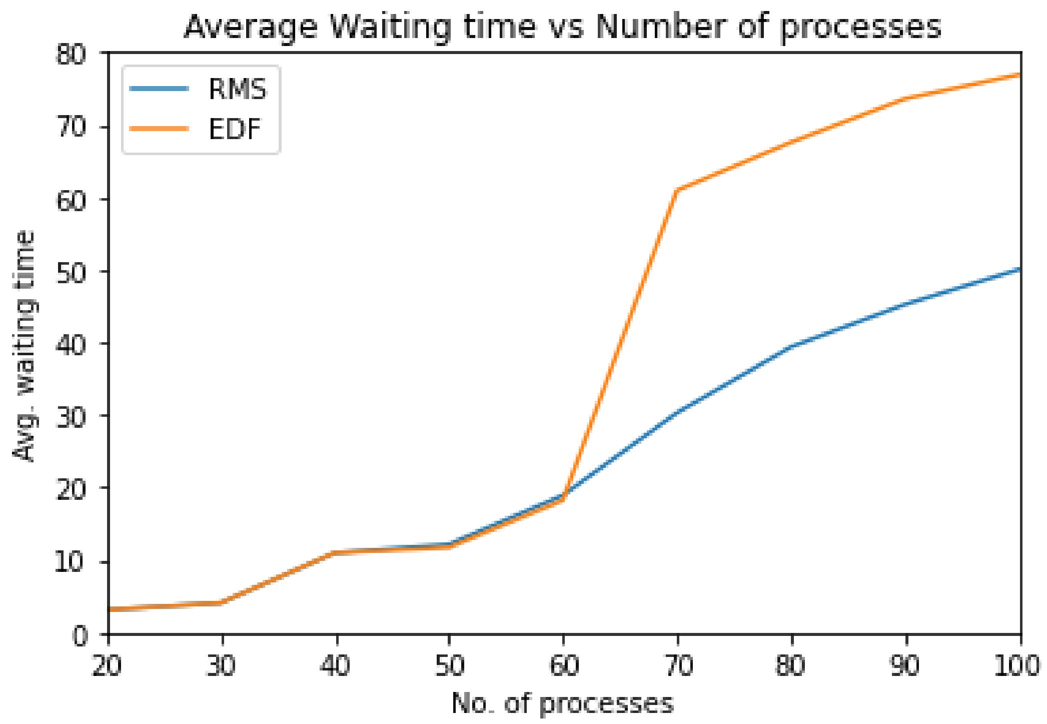
3. Comparison of performance RMS vs EDF

- Graph1



We see that initially there are no processes that miss their deadline, but as number of processes increase, number of processes that miss their deadline also increases. This is true for both algorithms. We also notice that as number of processes increases, RMS performs better than EDF. This is because of the observation that number of processes that miss their deadline is more in case of EDF with increase in number of processes.

- Graph 2



Initially, we see that RMS and EDF have same average waiting times. But as number of processes increase, average waiting time shoots up in case of both algorithms. We also notice that as number of processes increases, RMS performs better than EDF. This is because of the observation that average waiting time is more in case of EDF with increase in number of processes.

We can conclude that with increase in number of processes or system loading, RMS performs better than EDF.