Name: Aravinda Kumar Reddy Thippareddy

Roll No.: CS20BTECH11053

Course: OS-II

Assignment-3 REPORT

RMS (Rate Monotonic Scheduling):

- ➔ Rate-monotonic scheduling algorithm schedules periodic tasks using a static priority policy with preemption.
- ➔ If a lower priority process is running and a higher priority process becomes available to run, it will preempt the lower priority process.
- ➔ The shorter the period – the higher the priority; the longer the period – the lower the priority.

EDF (Earliest Deadline First Scheduling):

- ➔ EDF scheduling assigns priorities dynamically according to deadline.
- ➔ Under the EDF policy, when a process becomes runnable, it must announce its deadline requirements to the system.
- ➔ The earlier the deadline – the higher the priority; the later the deadline the lower the priority.

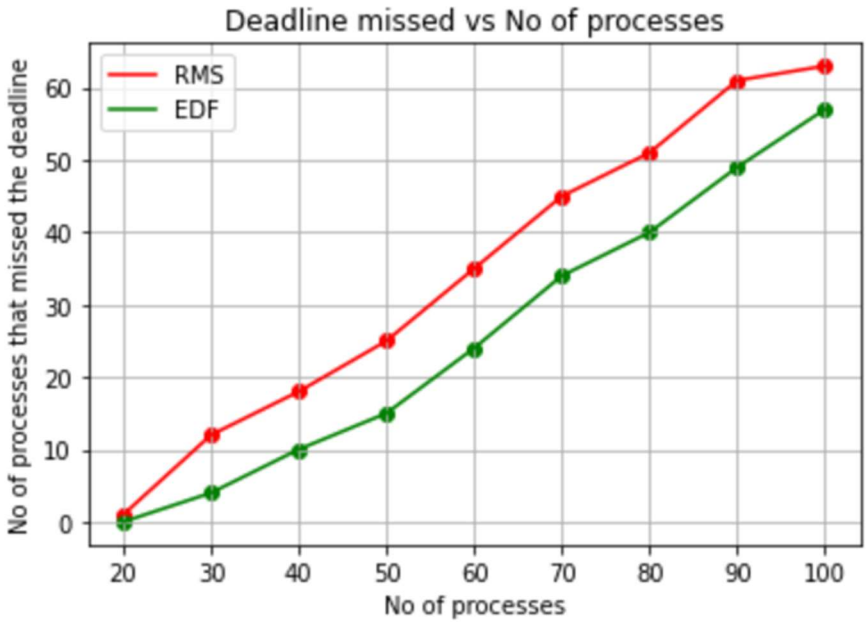
Design of my program:

- ➔ My design of the program is similar for both algorithms, except the priority condition of algorithms is different.
- ➔ I have read the contents of input file and stored the content into arrays.
- ➔ I created one array each for storing process id, period or deadline, number of times the process comes to the system, processing time. These arrays are left unchanged.
- ➔ I used other arrays, to update and store deadlines(dynamic_deadline), left processing time(dynamic_processing_tym), how many more times a particular process is yet to come to system(dynamic_k), to know whether the process is currently waiting to execute or not(dynamic_status).
- ➔ I have declared arrays to store average waiting time of each process, and also number of successful finishes of each process.
- ➔ I used an array(preemp_check_processing_time) to know if the process is resuming or starting or executing.
- ➔ I have declared a variable sum_k and stored sum of all values stored in k into sum_k.
- ➔ I had run a loop, till sum_k (decreased by 1 after each process completes or misses deadline) becomes 0.
    1. Each run of the loop will check whether to preempt a process or start/resume a process.

2. If we are to preempt a process then we will preempt the process, do necessary modifications and repeat the loop.
3. If we are to start/resume a process we will first find the process with the highest priority, and start or resume it based on the situation.
4. Then it will look for the nearest incoming process.
5. The executing process will be finished if the current process can be completed before the nearest incoming process comes, else the modifications will be done accordingly since the new process arrives.
6. We will terminate the process (when it is going to start), if we know that it will miss deadline and do modifications accordingly.
7. And by the end of this loop our log file will be ready.

➔ Finally, using the information stored in waiting times array and successful finishes array I have created the stats file.

GRAPH-1: Deadline missed vs Number of processes

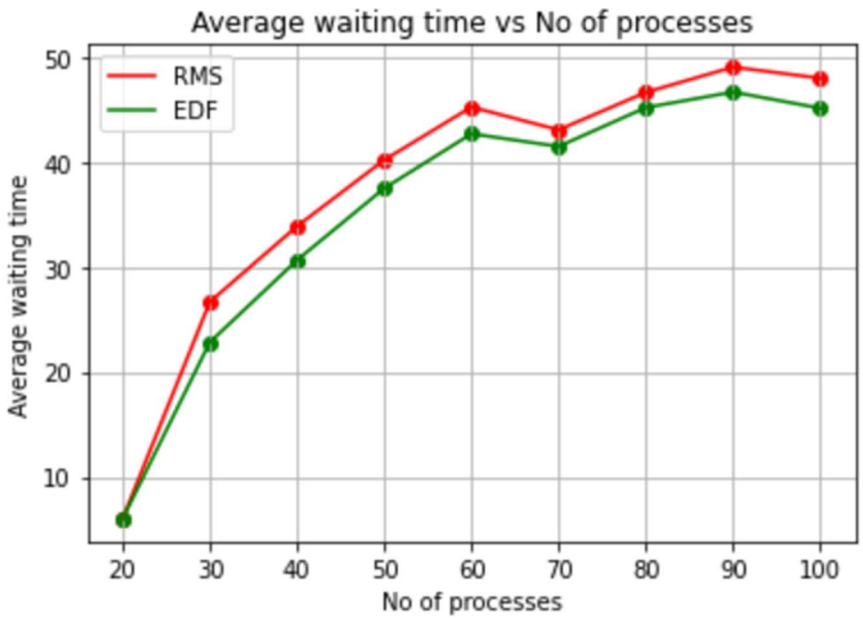


Observations:

➔ At any number of processes, number of processes missed deadline of RMS is greater than or equal to that of EDF.

➔ For both RMS and EDF, number of processes missed deadline kept on increasing with number of processes.

GRAPH-2: Average waiting time vs Number of processes



Observations:

- ➔ At any number of processes, average waiting time of RMS is greater than or equal to that of EDF.
- ➔ Average waiting times of both the curves raised from 0 to 60, decreased from 60 to 70, again increased from 70 to 90, and again decreased from 90 to 100.