Operating Systems 2 – CS3523

Programming Assignment 3 – Report

Suraj Telugu – CS20BTECH11050

Algorithms:

- **Rate Monotonic Scheduling Algorithm:** It schedules periodic processes using static priority policy with preemption for real time operating systems. Each process is assigned with a priority based on its period; shorter period has higher priority.

- **Earliest Deadline First Scheduling Algorithm:** It schedules periodic processes using dynamic priority policy with preemption for real time operating systems. It first selects a process according to its deadline such that a process with the earliest deadline has higher priority than others.

Design of the Program and Complications:

Rate Monotonic Scheduling Algorithm Simulator: (Assgn3-RMSCS20BTECH11050.cpp)

1. This Program uses RMS Algorithm to schedule the programs given in “inp-params.txt” and gives the output of simulation in “RMS-Log.txt” and the data obtained by simulation in “RM-Stats.txt”.

2. All the information of a particular process is stored in struct Process (INPUT) which serves as a Process Control Block (PCB) for the scheduling algorithm.

3. Memory is allocated for storing data of processes which is read from the given file, all the other values of PCB are initialised and stored in the INPUT struct array (prcs).

4. RMA_Sheduler function is used which takes input Number of processes and INPUT struct array (prcs). The algorithm is implemented in this function.

5. A struct Process_Priority (prcs_pr) is created which stores process ids and priorities of processes which are used for discrete event simulation. New struct is used to coordinate easily and to ensure the data in the prcs array do not get corrupt.

6. Two priority queues (min heaps) of struct prcs_pr are initialised and a comparison function is created which compares the priorities and arranges the process id in the form of min heaps. Min heaps are used because the process with lower priority (period) is given higher preference. Initially all processes are given 0 priority and they are arranged according to process id into min heap initially.

7. In those priority queues run_pq serves as the ready queue and next_pq serves as a queue which takes care of which process has to enter the ready queue.

8. A While loop is created which runs until both queues are emptied. The run_pq takes care processes that are ready and next_pq schedules which process has to enter run_pq according to its period.

9. At some time t, a process in the run_pq queue has 3 possible outcomes: Finishes its execution, Gets preempted by another process, misses its deadline.

10. If else conditions are written according to the events conditions, time and waiting time are updated in each event. The Log data is printed into “RMS-Log.txt” file.

11. For preemption 3 conditions are put to ensure the process does not preempt for every time it tries to run. Deadlines missed if process remaining burst time not equal to 0 and the conditions for missing a deadline is written accordingly.

12. Once the time reaches the period of a particular process it is again added into the ready queue until its number of reptations (k) equals 0. Thus, every process checks with all the if else conditions and enters its event and executes accordingly.

Earliest Deadline First Scheduling Algorithm Simulator (Assgn3-EDFCS20BTECH11050.cpp)

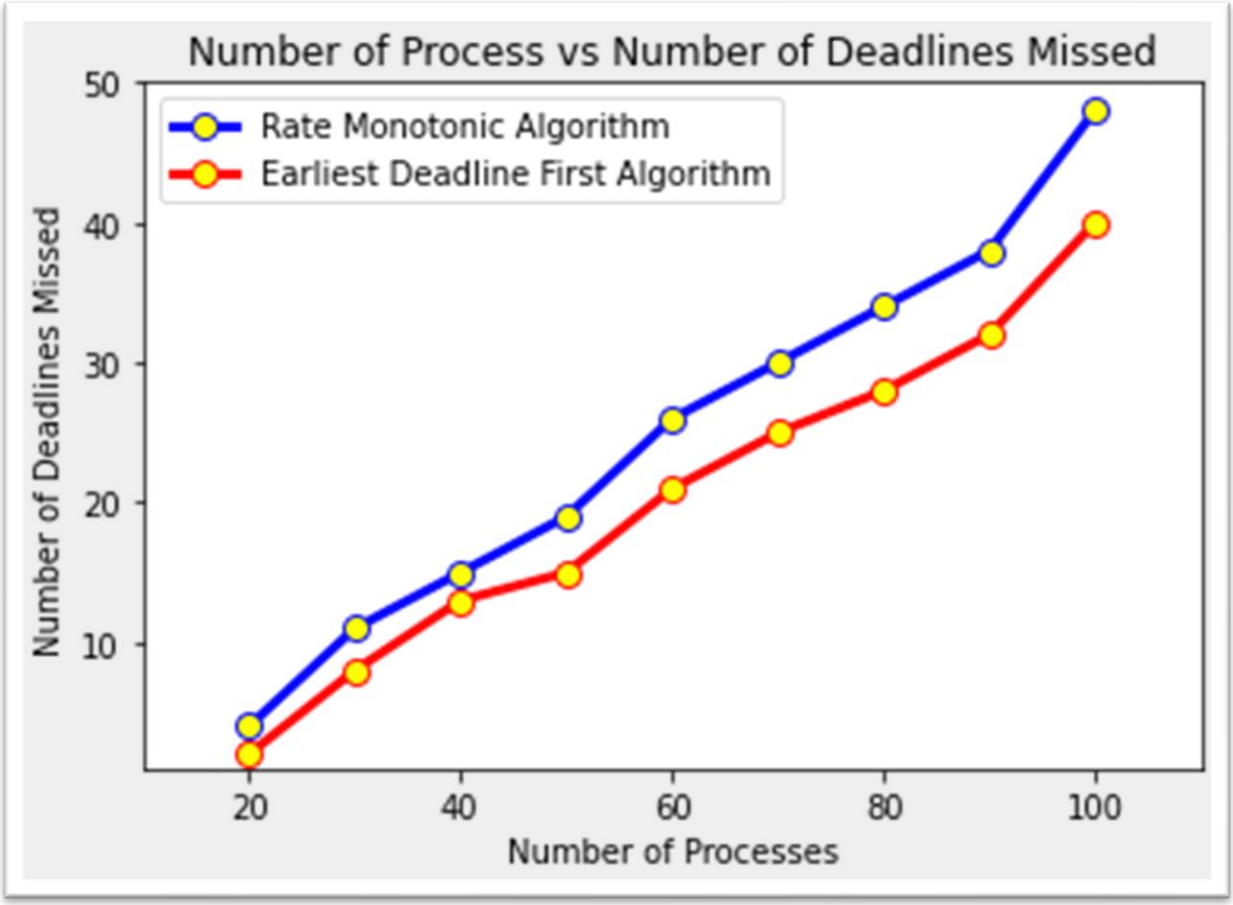
1. By making some changes to the code of Assgn3-RMSCS20BTECH11050.cpp we can get EDF Algorithm simulation output and data.

2. Input values are taken the same as RMA-Scheduling. EDF_Scheduler function is used where the priority queues now take priorities based on Next Deadline (next_dl) in both ready queue (run_pq) and next processing queue (next_pq).

3. The preemption condition is changed instead of periods next deadline are used. The condition of deadline missing is also changed. The condition for entering of process into ready queue is changed according to its deadline.

Graphs of Simulation and Analysis:

Graph 1: Total Number of processes vs Number of Deadlines Missed



Data Used for generating Graph:

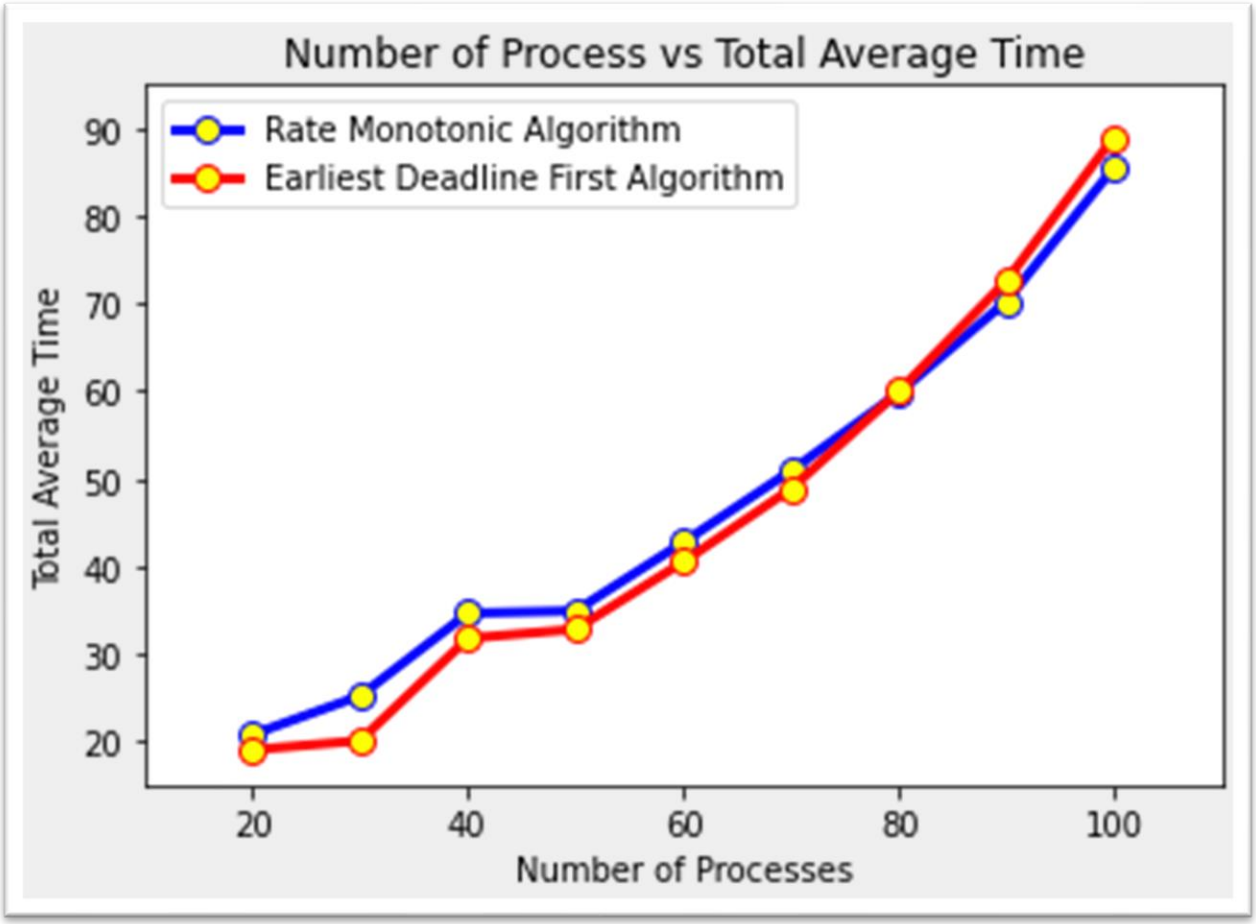
- 20 ≤ Number of Processes ≤ 100
- 10 ≤ Burst Time ≤ 100
- 50 ≤ Period ≤ 400
- Number of repetitions for each process = 10

Analysis and Conclusion:

- As the number of processes keep increasing for both algorithms number of processes whose deadline missed increased. EDF Algorithm has lesser number of deadlines missed processes compared to RMS Algorithm.

- As per above graph EDF algorithm has more successfully completed processes and lesser missed deadline processes, Thus EDF algorithm performs better than RMS algorithm.

Graph 2: Total Number of processes vs Total Average Time



Data Used for generating Graph:

- 20 ≤ Number of Processes ≤ 100
- 10 ≤ Burst Time ≤ 100
- 50 ≤ Period ≤ 400
- Number of repetitions for each process = 10

Analysis and Conclusion:

- Total Average waiting time of RMS and EDF are nearly same for given data and as the number of processes increases average waiting time increases for both algorithms.

- As per the graph obtained from data initially the average waiting time of RMS Algorithm is slightly greater than EDF Algorithm. For higher number of processes EDF Algorithm has more average waiting time compared to RMS Algorithm.

- For lower number of processes (CPU Utilisation < 1) EDF Algorithm performs better than EDF Algorithm, for higher number of processes (CPU Utilisation > 1) RMS Algorithm performs better than EDF Algorithm in terms of Average waiting time.