Operating Systems 2
SPRING SEMESTER 2018

INSTRUCTOR: DR. SATHYA PERI
Mayank Hooda -cs16btech11022@iith.ac.in

1. *Design of the Program :*

We use Discrete Event Simulation(DES) to simulate the schedule dispatcher. The scheduler does real-time scheduling with two different algorithms i.e. Rate Monotonic (RM) and Earliest Deadline First (EDF) .

Process :

We define a process by making a class ***Process*** which contains the standard parameters of process like processID , period , CPU burst time , deadline , number of repetitions . We then make a ***ready queue*** which contains all the processes which are waiting for their turn for the CPU. We also define some custom parameters like ***inQueueStatus*** (returns whether process is in ready queue or not) , deadline of process, time left for finishing . Effective total number of processes would be :

$$\sum_k P[k].numberofRepitions$$ where P[k] is kth process.

Rate Monotonic(RM) :

Rate Monotonic Scheduling is characterised by giving priority to processes which have the least period. This is static-priority scheduling as period of a process is constant throughout execution.

Earliest Deadline First (EDF):

Earliest Deadline First Scheduling is characterised by giving priority to processes which have the earliest deadline. This is a dynamic-priority scheduling as the deadline of a processes is increased by its period after every repetition.

Comparison parameters :

We have the following metrics for comparing the performance of the two algorithms :

1. **Deadline Miss :**

A process misses a deadline if before completion of execution of the process, the deadline expires .

2. Waiting Time :

Waiting Time is defined as the time the process spends in ready queue not executing on CPU.

3. Turnaround Time :

Turnaround time is defined as the total time elapsed between completion of a process and its arrival.

The better algorithm would be the one in which all three parameters are minimised.

2. Working of the Program :

In DES, at each second we check the arrival of processes.We keep a variable of type Process *running* .*running* has a member *on_cpu* which contains whether any process is running on CPU at that time.

We define priority of process in RM as one with least period whereas in EDF it is defined as the one with least deadline.

At any given time *t* , a process enters the ready queue if *t*%(P[i].priority) = 0 , where P[*i*] is *i* th process. If such a process is already in queue , then it misses the deadline and we kill the process and increase it's deadline by its period. If *running* ==P[i] then we kill that process too and hence no process will be on CPU at that time. Now in such a situation if the ready queue is not empty we pick the element with highest priority from the queue and put it on CPU. Else , if running *on_cpu* is *false* and ready queue is empty then CPU will be idle until another process comes into queue at its periodic interval.

Now we compare the priority of *running* with that of the highest prior element of the queue , if its not empty and pre-empt the *running process* if it's priority is more.

Now we decrement the time_left of the running process at each iteration until it is greater than zero. If it is zero , then we mark that process as completed and increase the deadline of this process by its period and mark the *on_cpu* member of *running as false.*

We use a vector to represent a priority queue and store the addresses of processes in the queue such that change in the Process object would reflect in the queue.

Finally we find the average waiting time by incrementing it for a process at each step if it is in queue and not on CPU.

Average Turnaround time would be simply calculated as subtracting the time between end of a process , either successful or killed.

3. Graphs :

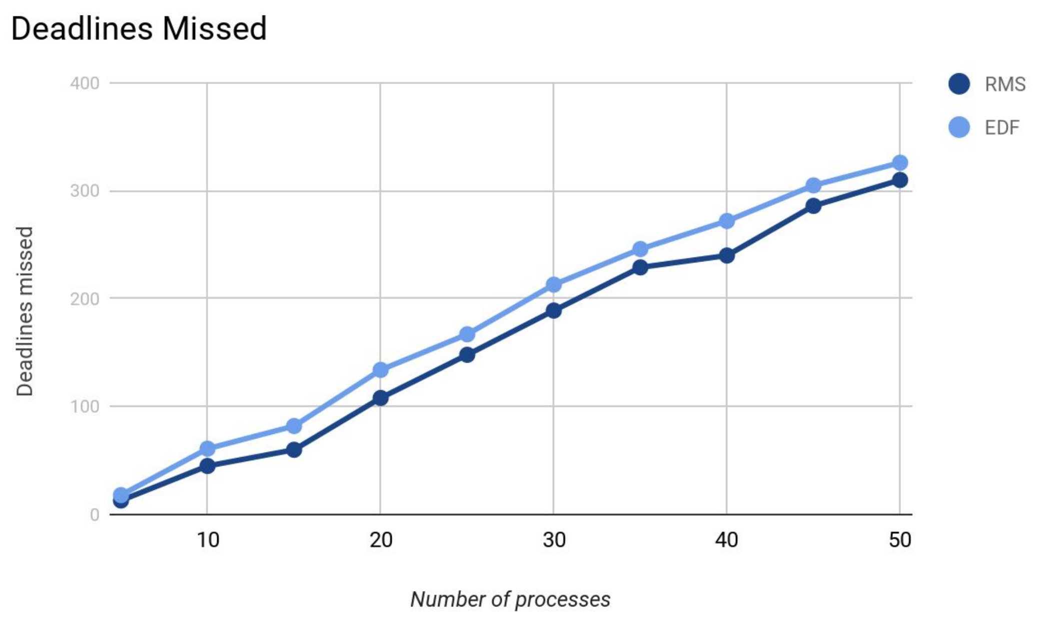
We use a c++ script for obtaining random parameters for n processes. For the two schedulings , the constraints we put on parameters are :

10 <= T (running time) < 30

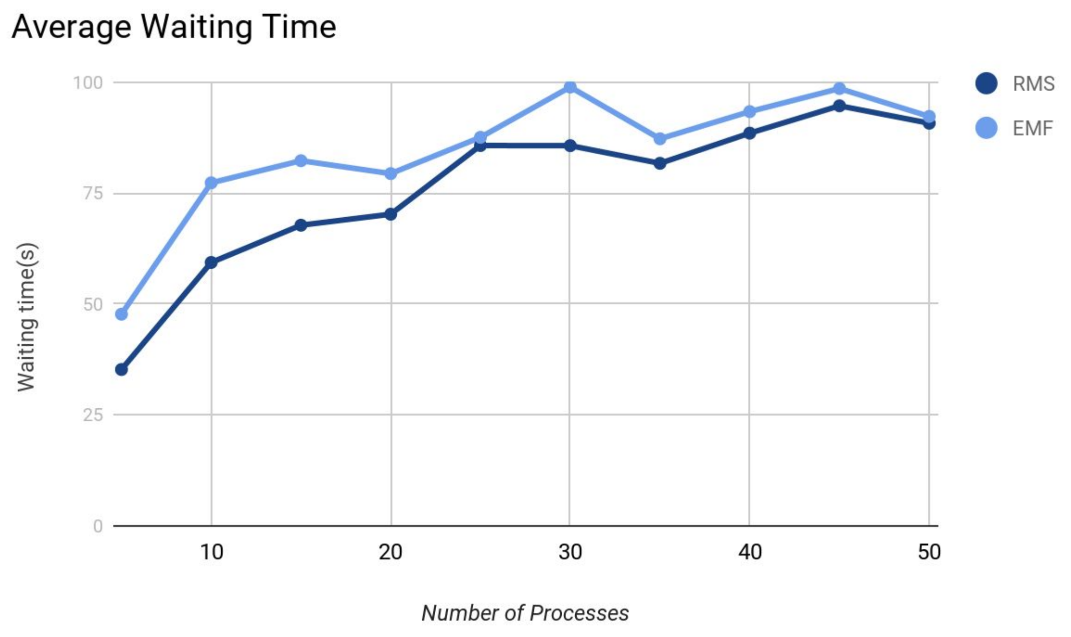
60 <= P (period) < 110

5 <= k (Number of repetitions) < 10

Graph 1 : Deadline missed vs Number of Process



Graph 2 : Average Waiting Time vs Number of Process



We can deduce from the graph that as N approach very large value , waiting time becomes constant and doesn’t rise with increasing N .

And RMS shows better performance than EDF , both in terms of waiting time , as well as fewer processes with exceeded deadline.