TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

Central Department of Computer Science and Information Technology Kirtipur, Kathmandu

Assignment
Advanced Operating System

Submitted By
Rishav Acharya
Roll no.: 01/2077

Submitted To
Mr. Ananda Kumar Shah

Real Time Systems

A real time system is a computer system that requires not only that the computing results be "correct" but also that the results be produced within a specified deadline period. The results produced after the deadline has passed - even if correct may be of no real value.

Some scheduling algorithms that address the deadline requirements are:-

i) Rate Monotonic Scheduling
ii) Earliest Deadline First Scheduling
iii) Proportional Share Scheduling
iv) Pthread Scheduling

i) Rate Monotonic Scheduling

Rate monotonic scheduling algorithm is a priority algorithm that belongs to static priority scheduling category of Real Time Operating Systems. It is preemptive in nature. Here in this algorithm priority is determined by comparing the period of given task or tuples. The tuple which has less period has higher priority & which has high period has low priority. If a lower priority process is running & a higher priority process becomes available to run it will preempt the lower priority process.

Example: Let us consider three tasks with period and execution time

$T_1(4,1)$ $T_2(5,2)$ $T_3(20,5)$

Here, priority is $T_1 > T_2 > T_3$

In this example T_1 is repeated every 4 unit of time for only 1 unit of time, T_2 is repeated every 5 unit of time for 2 unit of time & T_3 is repeated every 20 unit of time for 5 unit of time.

Since, T_1 has highest priority it starts at 0 unit of time and executes until 1 unit of time then, T_2 has higher priority & excutes for 2 unit of time until 3 unil of time & then, T_3 comes but executes only 1 unit of time because T_1 is repeated on every 4 unit of time. So, T_1 executes until 5 unit of time & T_2 runs until 7 unit of time. Again T_3 only executes for 1 unit of time until 8 unit of time Then, T_1 comes & executes until 9 unit of time. Though T_2 has higher priority than T_3, T_2 only repeats with interval of 5 unit of time So, T_3 executes here & T_3 preempts & T_2 executes from 10 to 12 unit of time. T_1 again repeats at 12 unit of time & executes until 13 unit of time. Since, T_2 already executes 2unit of times here comes T_3 & it executes 2unit of time until 15 unit of time & T_2 comes. T_2 only executes 1 unit of time because higher priority T_1 comes at 16 unit of times & executes. Then, again T_2 executes it's remaining 1 unit of time until 18 unit of time. T_3 has already completed its execution time for 20 unit of time & other tasks are also repeated at 20 unit of time. 18, 19 & 20 unit of time the processor remains idle.

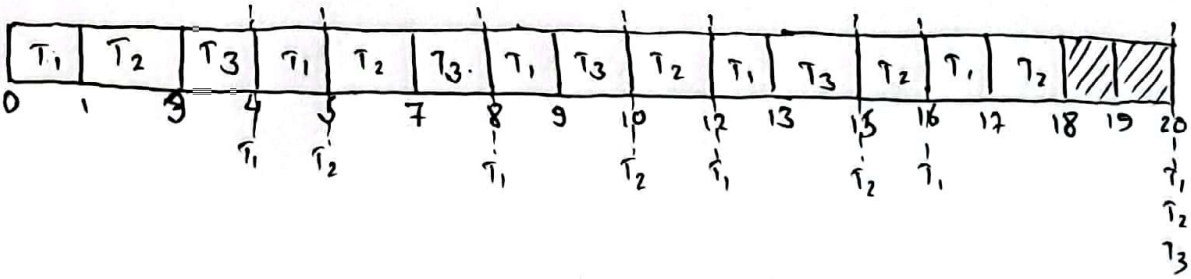

Figure: Rate monotonic Scheduling

iii) Earliest Deadline first Scheduling

EDF is an optimal dynamic priority scheduling algorithm used in real time system. It can be used for both static & dynamic real-time systems. It can schedule jobs using priority of the tasks. It assigns priorities to the task according to the absolute deadline. The task whose deadline is closest gets the highest priority. It is very efficient as compared to other scheduling algorithms in real time systems.

Example: Lets take an example of two tasks with period, execution time & deadline.

$T_1(2, 0.9, 2)$ $T_2(5, 2.3, 5)$

Here, T_1 is executed at every 2 unit of time & T_2 at every 5 unit of time. At first the deadline of T_1 is less than the deadline of T_2. So, T_1 executes first with it's execution time 0.9 then T_2 is executed for 1.1 unit of time & ends at 2 unit of time because again T_1 is executed first as it has it's deadline 4 which is less than the deadline of T_2 i.e. 5. T_1 is executed upto 2.9 unit of the time & then, T_2 is executed upto 4 unit of time. Again, the deadline is checked & T_1 has at 6 unit of time but T_2 has at 5. So, T_2 is executed for 0.1 unit of it's remaining execution time Now, T_1 is executed upto 5 unit of time. At 5 unit of time, T_1 is executed 0.9 of its executed time. Since, T_1 is executed once until 6 unit of time, but T_1 already executed from 4.1 to 5 unit of time so, T_2 is executed 1 unit of its 2.3 unit execution time. At 6 unit of time, the deadline is checked again & the deadline of T_1 is 8 & T_2 is 10. So, T_1 is executed till 6.9 & then T_2 to 8. At 8 unit of time, the deadline of both tasks is 10 since, T_1 is executed first with execution time 0.9 upto 8.9 unit of time & then T_2 is executed until 9.1 unit of time. Now, the remaining unit of time the processor remains idle.

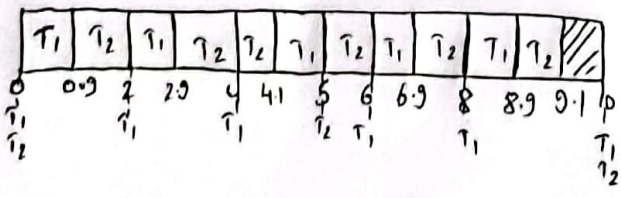

Figure: EDF Scheduling

iii) Proportional Share Scheduling

Proportional Share Schedulers operate by allocating T shares among all applications. An application can receive N shares of time, thus ensuring that the application will have N/T of the total processor time. As an example assume that there is a total of T=100 shares to be divided among three processors: A, B & C.

A is assigned 50 shares, B - 15 shares & C is assigned 20 shares. This scheme ensures that A will have 50 percent of total processor time, B will have 15 percent & C will have 20 percent.

Proportional Share schedulers must work in conjuction with an admission control policy to guarantee that an application receives its allocated shares of time. An admission control policy will only admit a client requesting a particular number of ~~str~~ shares if there are sufficient shares available. In our example, ~~othe~~ we have allocated

$$50 + 15 + 20 = 75 \text{ shares}$$

of the total of 100 shares. If a new process D requested 30 shares, the admission control would deny the entry of D into the system.