Earliest Deadline First (EDF) Task Scheduling

We are given four independent pre-emptive tasks to be scheduled using the Earliest Deadline First (EDF) priority algorithm. In EDF, the task with the nearest deadline is executed first. If multiple tasks have the same deadline, we break ties using Rate Monotonic (RM), which gives higher priority to the task with the shorter period. Here's how we approached the problem.

# Step 1: Determine Hyperperiod

The hyperperiod is the least common multiple (LCM) of all the task periods, which is the interval over which the schedule repeats. For the given tasks:  
Task periods: 30, 40, 60, and 120.  
The LCM of these periods is 120. Therefore, the schedule repeats after every 120 time units.

# Step 2: Task Characteristics and Arrivals

The tasks' key parameters are summarized in the table below. Each task arrives periodically and has a specified worst-case execution time (WCET) and deadline.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Period (Pi) | WCET (Ci) | Deadline (Di) | Initial Delay (Φi) |
| T1 | 30 | 10 | 30 | 0 |
| T2 | 40 | 10 | 40 | 0 |
| T3 | 60 | 10 | 50 | 0 |
| T4 | 120 | 15 | 100 | 0 |

# Step 3: Task Scheduling Using EDF

We use the EDF algorithm to determine the task execution schedule between time t = 0 and t = 120. Tasks are executed based on their earliest deadlines. When two tasks have the same deadline, the tie is broken using Rate Monotonic prioritization, favoring tasks with shorter periods.  
The detailed schedule is as follows:

|  |  |
| --- | --- |
| Time Interval | Task Executed |
| t = 0 - 10 | T1 |
| t = 10 - 20 | T2 |
| t = 20 - 30 | T3 |
| t = 30 - 40 | T1 |
| t = 40 - 50 | T2 |
| t = 50 - 65 | T4 |
| t = 65 - 70 | T4 |
| t = 70 - 80 | T1 |
| t = 80 - 90 | T3 |
| t = 90 - 100 | T1 |
| t = 100 - 110 | T2 |
| t = 110 - 120 | T4 |

# Summary

In the interval [0, 120), tasks are scheduled based on their deadlines, and the schedule repeats after 120 time units. The earliest deadline determines which task runs, with ties broken by Rate Monotonic prioritization.