8. Priority Schedule (RTAI ✓)

- CPU is allocated to the process with highest priority.
- Equal priority jobs are scheduled FCFS or in a Round Robin fashion.
- Preemptive or non-preemptive.
- Shortest job is a special case or PS.
- Potential problems: Blocking or Starvation.
a) Rate Monotonic

- Based on execution period of each task
- Shorter period $\rightarrow$ higher priority
- Static (fixed) policy: priorities remain fixed
- Preemptive
- Task deadline assumed equal to task period

$\text{Schedulability: } \sum_{i=0}^{n-1} \frac{e_i}{p_i} \leq n \left( \frac{1}{2^n} - 1 \right)$

CPU utilization bound

$e_i = \text{execution time}, \quad p_i = \text{period of task}$

(Sufficient condition)

Examples 1a, 1b
a) Rate Monotonic (cont.)

– The CPU utilization bound $\rightarrow \approx 0.69$ as $n \rightarrow \infty$

  Just for fun: can you prove it?

– In practice, a typical bound $= 69\% - 88\%$ (relaxed)

**Critical Zone Theorem:** if computed utilization $\leq$ bound, then the system is guaranteed to meet all task deadlines in all task orderings.

– If a rate monotonic schedule does not exist, no other fixed rate scheme will work.

– This algorithm is **stable:** if lower priority tasks are added $\Rightarrow$ the higher priority ones remain schedulable.
b) Earliest Deadline First

- Dynamic priorities = closest deadline
- “Closest deadline” must be assessed “on-the-fly”.
- Schedulable if sum of task loading (CPU util.) ≤ 100% (Necessary and sufficient condition).
- This algorithm is not stable: if new tasks are added and load > 100% ➞ it’s not possible to predict which tasks will fail.

Example 2
L6: Example 3 (Earliest Deadline First)

P1: \( e_1 = 1, p_1 = 8 \);  
P2: \( e_2 = 2, p_2 = 5 \);  
P3: \( e_3 = 4, p_3 = 10 \)

Units of time may be considered to be schedulable time slices.

CPU utilization: \[ \frac{1}{8} + \frac{2}{5} + \frac{4}{10} = \frac{37}{40} = 0.925 < 1 \rightarrow \text{Schedulable!} \]

- At time = 5, both P2 and P3 have the same deadline. EDF may schedule either one.
- Notice the 20\(^{th}\) time slice (starting at \( t = 19 \)): CPU is not being used.
- The least common multiple of the periods is 40. The scheduling pattern can repeat every 40 units of time. Only 37 of those 40 are used by P1, P2 or P3.
- What if there are no time slices? Would the schedule be any different?
c) Least Laxity

- Laxity = Deadline – Execution Time  (*Dynamic!*)
- Smaller laxity $\rightarrow$ higher priority
- Prior knowledge of execution times (or at least of upper bounds) required.
- Same schedulability test as ED algorithm.
- Easy to work in systems with hard and soft deadlines.
- Not stable.
- Tasks with execution time $> \text{deadline}$ are not scheduled (in earliest deadline they would be).

Example 4
L6: Scheduling Algorithms (cont.)

d) **Maximum Urgency**

- Initially, priorities are assigned as in rate monotonic.
- Tasks are divided into critical / non-critical.
- Least laxity is applied to the critical set.
Assignment

• Read Section 12.5.1

• Homework #1 (posted on Blackboard)