1. Deadline Monotonic Scheduling (DMS)

Theoretical Explanation

The deadline monotonic scheduling is a scheme for scheduling tasks in the realtime system according to the deadline. Basically the straightforward one, tasks with earlier deadlines have higher priority. It makes sense because we want to first finish those necessary tasks.

Once priorities are assigned at the beginning, they don't change in the course of execution in DMS. Assuming Task A deadline is 5 seconds, Task B deadline is 10 seconds, Task A has higher priority than Task B.

DMS works like this:

- Look at all the tasks and their deadlines
- Give highest priority to the task with the shortest deadline
- Give next highest priority to the task with the next shortest deadline
- And so on…

RMS and DMS are actually the same when deadlines equal periods (how often a task repeats). The thing is that, unlike DMS, in which deadlines may be shorter than a period, DMS allows handling such cases.

Example

Let's say we have three tasks:

- Task 1: Execution time = 1, Period = 8, Deadline = 6
- Task 2: Execution time = 2, Period = 12, Deadline = 10
- Task 3: Execution time = 3, Period = 16, Deadline = 14

Using DMS, the priorities would be:

1. Task 1 (highest priority)
2. Task 2 (middle priority)
3. Task 3 (lowest priority)

This ensures that tasks that need to finish sooner get CPU time first.

2. Least Laxity First (LLF)

Theoretical Explanation

While Least Laxity First changes priorities while tasks are running, DMS does not. It makes decisions when to run the next task based on something called laxity.

Specifically, laxity is how much time a task can wait, before it must start running by a deadline absolutely. We calculate it as:

Laxity = (Deadline - Current time) - Remaining execution time

The highest priority is given to the task with the smallest laxity. This suggests priorities can alter throughout the run duration as time passes and work retains up.

Zero laxity means that a task has to be run without break whenever it is scheduled and no delay can happen or it will be late, as in case of zero slack.

Example

Imagine two tasks at time t = 0:

- Task A: Execution time = 3, Deadline at t = 7
- Task B: Execution time = 5, Deadline at t = 8

Initial laxity calculations:

- Laxity of Task A = (7 - 0) - 3 = 4
- Laxity of Task B = (8 - 0) - 5 = 3

Since Task B has lower laxity (3 < 4), it gets scheduled first.

3. Minimal Slack Scheduling (MSS)

LLF is essentially the same as Minimal Slack Scheduling. In fact, some textbooks or papers can use different formulas or different terms, however the basic idea is the same - it will take a huge task and rank it in consideration of how urgently it must finish before the deadline.

Both LLF and MSS, as the main idea, place the tasks with no flexibility in scheduling with the focus. Both algorithms recompute priorities continuously with time passing.

Since MSS and LLF are practically functionally equivalent, we do not need to speak about MS separately from LLF.

Comparative Analysis

Advantages and Limitations

Deadline Monotonic Scheduling (DMS)

Advantages:

- Simple to understand and implement
- Low overhead - priorities are set once
- Predictable behavior
- Good for simple systems with periodic tasks

Limitations:

- Not optimal - might fail even when a workable schedule exists
- Can't adapt to changes during execution
- Doesn't consider actual execution times

Least Laxity First (LLF/MSS)

Advantages:

- Can find a working schedule if one exists (optimal)
- Adapts to actual execution times
- Better CPU utilization
- Handles mixed task types well

Limitations:

- High computational overhead from constant recalculations
- Frequent task switching when laxities are similar
- Complex to implement
- Less predictable than DMS

When to Use Each Algorithm

Use DMS when:

- You have a simple system with regular tasks
- You need to minimize overhead

- Predictability is more important than squeezing out maximum performance
- You're working with limited computing resources

Use LLF/MSS when:

- Meeting all deadlines is absolutely critical
- Task execution times vary a lot
- You have enough computing power to handle the overhead
- You need to maximize CPU utilization

Real-world Considerations

In real life, the choice depends on what we're're building:

1. DMS may be better if predictability is important for safety critical systems like medical devices or car brakes.

2. In the presence of task patterns that change though deadlines cannot be missed, LLF/MSS might be overkill, but only at some expense.

3. Combinations or variations of these algorithms are used in many practical systems that want to combine getting the best processing results with fitting the size and performance constraints.

Conclusion

However, DMS is simple and predictable, but not always optimal. It gives fixed priorities according to deadlines. MSS is less complex but also less adaptable: instead of making continual changes to task priorities depending on the urgency of various tasks, it stops after it has determined which tasks will run first, second, etc.

What choice we make depends on your system requirements, dependence on simplicity, predictability or using all resources that are available. It helps one understand better how to design systems for their timing needs, they can use resources efficiently.