1.4.3. Shortcomings of EDF

In this subsection, we highlight some of the important shortcomings of EDF when used for scheduling real-time tasks in practical applications.

Transient Overload Problem:

Transient overload denotes the overload of a system for a very short time. Transient overload occurs when some task takes more time to complete than what was originally planned during the design time. A task may take longer to complete due to many reasons. For example, it might enter an infinite loop or encounter an unusual condition and enter a rarely used branch due to some abnormal input values. When EDF is used to schedule a set of periodic real-time tasks, a task overshooting its completion time can cause some other task(s) to miss their deadlines. It is usually very difficult to predict during program design which task might miss its deadline when a transient overload occurs in the system due to a low priority task overshooting its deadline. The only prediction that can be made is that the task (tasks) that would run immediately after the task causing the transient overload would get delayed and might miss its (their) respective deadline(s). However, at different times a task might be followed by different tasks in execution. However, this lead does not help us to find which task might miss its deadline. Even the most critical task might miss its deadline due to a very low priority task overshooting its planned completion time. So, it should be clear that under EDF any amount of careful design will not guarantee that the most critical task would not miss its deadline under transient overload. This is a serious drawback of the EDF scheduling algorithm.

Resource Sharing Problem:

When EDF is used to schedule a set of real-time tasks, unacceptably high overheads might have to be incurred to support resource sharing among the tasks without making tasks to miss their respective deadlines. We examine this issue in some detail in the next lesson.

Efficient Implementation Problem:

The efficient implementation that we discussed in Sec. 3.4.2 is often not practicable as it is difficult to restrict the number of tasks with distinct deadlines to a reasonable number. The efficient implementation that achieves O(1) overhead assumes that the number of relative deadlines is restricted. This may be unacceptable in some situations. For a more flexible EDF algorithm, we need to keep the tasks ordered in terms of their deadlines using a priority queue. Whenever a task arrives, it is inserted into the priority queue. The complexity of insertion of an element into a priority queue is of the order log2 n, where n is the number of tasks to be scheduled. This represents a high runtime overhead, since most real-time tasks are periodic with small periods and strict deadlines.

<https://nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Embedded%20systems/Pdf/Lesson-30.pdf>