Review
Guangmo Tong

Abstract—This is the review of two papers, Tardiness Bounds under Global EDF Scheduling on a Multiprocessor (RTSS’05) and An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems (RTSS’12).

1 Tardiness Bounds under Global EDF Scheduling on a Multiprocessor

1.1 Topics and Ideas
The major topic in this paper is the scheduling of soft real-time sporadic task systems on a multiprocessor. This paper derives the tardiness bound under preemptive or non-preemptive global earliest-deadline-first (EDF) schedule when the total utilization is less than or even equal to the number of processors. The conclusion shows that if we are able to tolerant the bounded tardiness then the resource can be fully utilized in the long run.

1.2 Intellectual merit
To derive the tardiness, this paper compares the resource allocation to jobs between the EDF schedule and the processor sharing (PS) schedule. PS schedule can be a reference schedule in that in PS schedule every job is able to finish its execution exactly at its deadline. Therefore the difference of allocation represents the workload that remains to be done. Another important method used in this paper is induction. To derive the tardiness of a job, \( T_{i,j} \), we assume that jobs whose deadlines are earlier than the deadline of \( T_{i,j} \), \( d_{i,j} \) have given bounded tardiness. With this assumption, we are able to derive the tardiness of job \( T_{i,j} \) if the difference of allocation till \( d_{i,j} \) is not more than the lower bound. The details can be found in the paper.

1.3 Strength
This paper considers the soft real-time sporadic task systems on multiprocessor under the preemptive and non-preemptive global EDF. The strength of this paper can be embedded in following facts. First, Multiprocessor-based design has been used widely in the present-day applications. For example, virtual-reality systems, systems that track people and machines, and signal-processing systems such as synthetic-aperture. In this case, the previous theories on uniprocessor are insufficient. Then, not all real-time systems require that the deadline of job has to be meet. In other words, such as in multimedia and gaming systems, we can tolerant tardiness as long as it is bounded. Finally, this paper derives a tardiness bound under non-preemptive global EDF, which provides a theory foundation for the case in practice that tasks cannot be paused before completion.

2 An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems

2.1 Topics and Ideas
This paper considers soft real-time self-suspending task systems under the preemptive global EDF and derives a new schedulability test with \( O(m) \) suspension-related utilization loss. Similar to the first paper, this paper also analyze the tardiness bound by using the allocation difference between PS schedule and EDF schedule.

2.2 Intellectual merit
In self-suspending task system jobs alternate between computation and suspension phases,
which is the main difference between the system models in two papers. When analyze self-suspending systems, if we take all the suspensions as executions obviously as in the previous work, we are able to derive the tardiness bound by following the framework in the first paper. However, this will restrict the total utilization of the system and renders an \(O(n)\) utilization loss. This paper works out that to avoid the worst-case scenario, by doing some transformation of the schedule, converting at most \(m\) tasks’ suspension into computation is enough. In this case, the utilization loss can be reduced to \(O(m)\) with the improved schedulability test.

### 2.3 Strength

The strength of this paper can be seen in the following aspects. First, contrast to the first paper, this paper considers job with not only execution phase but also suspension phases. As the paper says, suspension delays may occur in many systems due to the cases such as the interaction with external devices and resource sharing. Then this paper derives an tardiness bound with \(O(m)\) utilization loss. This result is much better than the previous result in that when the number of tasks is very large then the \(O(n)\) utilization loss can be highly significant.

### References
