CHAPTER 4

Recent Results in Real-Time Scheduling

R. Bettati, D. Gillies, C. C. Han, K. J. Lin
C. L. Liu, J. W. S. Liu and W. K. Shih

Department of Computer Science
University of Illinois at Urbana–Champaign
Urbana, Illinois 61801

Abstract

This paper gives an overview of several recent results on real-time scheduling. Specifically, it presents the workload models that characterize the following types of tasks: tasks with AND/OR precedence constraints, tasks with temporal distance constraints, distributed tasks with overall deadlines, and tasks with deferrable deadlines. Suitable algorithms for scheduling tasks of each type are described.

INTRODUCTION

A (hard) real-time (computing) system is one in which some computations are time-critical. Each time-critical computation has a deadline; the computation must be executed to completion by the deadline so that a result can be made available in time. A result that is produced too late is of little or no use. Hence, the primary objective of real-time scheduling is to find schedules in which all deadlines are met whenever such schedules exist.

This paper gives an overview of some of our recent results on real-time scheduling. Specifically, it presents four problems: scheduling tasks with AND/OR precedence constraints, tasks with temporal distance constraints, distributed tasks with overall deadlines, and tasks with deferrable deadlines. Here, by a task, we mean a granule of computation that is treated by the operating system as a unit of work to be scheduled and executed.

We introduce now the notations and terms that will be used in the later sections. All the workload models described in this paper are extensions or variations of the following well-known, deterministic task model. This model characterizes the work to be scheduled and
executed as a task system consisting of $n$ tasks. We denote the task system by $T = \{T_1, T_2, \ldots, T_n\}$. Each task $T_i$ in $T$ has the following parameters:

1. **ready time** $r_i$ after which $T_i$ can be scheduled for execution,
2. **deadline** $d_i$ by which $T_i$ must be completed, and
3. **processing time** $p_i$ that is required to execute $T_i$ to completion.

We say that a task has a deadline when its deadline is finite and has no deadline when its deadline is infinite. The time interval $[r_i, d_i]$ is called the **feasibility interval** of $T_i$.

The tasks in a task system $T$ may be dependent on each other. Their dependencies, called **precedence constraints**, are specified by a partial order relation $<$ defined over the set $T$. $T_i < T_j$ if the execution of $T_j$ cannot begin until $T_i$ has completed. $T_i$ is said to be a **predecessor** of $T_j$, and $T_j$ a **successor** of $T_i$, if $T_i < T_j$. $T_i$ is an **immediate predecessor** of $T_j$ if there is no other task $T_k$ such that $T_i < T_k < T_j$; in this case $T_j$ is an **immediate successor** of $T_i$. We sometimes represent a task system by a directed graph. The vertex set of this graph is the set $T$, and there is an edge from $T_i$ to $T_j$ in the edge set of this graph if $T_i$ is an immediate predecessor of $T_j$. We refer to this graph as the **dependency graph** of the task system $T$.

A schedule of a task system $T$ on $m$ identical processors is an assignment of the tasks in $T$ to the processors such that at any time (1) each processor is assigned at most one task and (2) each task is assigned to at most one processor. A task is said to be **scheduled** (and **executing**) in an interval if it is assigned to a processor in the interval. A time interval during which no task is assigned to a processor is an **idle interval** on the processor. The total length of all the time intervals in which a task $T_i$ is scheduled is called the **assigned processor time** $\sigma_i$ of the task. A schedule is a **valid** one if no task is scheduled before its ready time, the assigned processor time of every task is equal to its processing time, and all the precedence constraints are satisfied. Given a schedule, we call the earliest and latest time instants at which a task $T_i$ is scheduled the **start time** $s_i$ and the **finishing time** $f_i$ of the task, respectively. A valid schedule is a **feasible schedule** if the finishing time of every task is at or before its deadline. A schedule is **non-preemptive**