ELSEVIER

Journal of Systems Architecture 46 (2000) 991–1004

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

Fixed priority scheduling of tasks with arbitrary precedence constraints in distributed hard real-time systems

Romulo Silva de Oliveira [a,*], Joni da Silva Fraga [b]

[a] II-Univ. Fed. do Rio Grande do Sul, Caixa Postal 15064, 91501-970 Porto Alegre-RS, Brazil
[b] LCMI/DAS-Univ. Fed. de Santa Catarina, Caixa Postal 476, 88040-900 Florianopolis-SC, Brazil

Received 29 January 1998; received in revised form 21 July 1999; accepted 21 September 1999

Abstract

This paper considers the schedulability analysis of real-time distributed applications where tasks may present arbitrary precedence relations. It is assumed that tasks are periodic or sporadic and dynamically released. They have fixed priorities and hard end-to-end deadlines that are equal to or less than the respective period. We develop a method to transform arbitrary precedence relations into release jitter. By eliminating all precedence relations in the task set one can apply any available schedulability test that is valid for independent task sets. © 2000 Published by Elsevier Science B.V. All rights reserved.

Keywords: Scheduling; Real-time systems; Fixed priority; Precedence relations

1. Introduction

Deadlines are critical in a hard real-time system. In this case it is necessary to have a pre-runtime guarantee that every task will always meet its deadline. Such pre-runtime guarantee can be attained with fixed priority scheduling [11]. Tasks receive priorities according to some policy and a schedulability test is used off-line to guarantee that all tasks will meet their deadlines in a worst-case scenario. At runtime, a pre-emptive scheduler selects the next task to run based on their fixed priorities.

Real-time scheduling in distributed systems is usually divided into two phases: allocation and local scheduling. Initially each task is assigned to a specific processor. The original allocation is permanent since it is usually not possible to migrate tasks at runtime. The second phase analyzes the schedulability of each processor.

Precedence relations are common in distributed systems. They are created by the necessity of synchronization and/or transfer of data between two tasks. It is possible to use offsets [2] to implement precedence relations. By defining an offset between the releases of two tasks it is possible to guarantee that the successor task will start its execution only after the predecessor is finished. This technique is sometimes called “static release of tasks” [14].

It is also possible to implement precedence relations by having the predecessor task to send a message to the successor task. It also informs the scheduler that the predecessor task has finished and the successor task can be released. This technique is sometimes called “dynamic release of

[*] Corresponding author.

E-mail addresses: romulo@inf.ufrgs.br (R.S. de Oliveira), fraga@lcmi.ufsc.br (J. da Silva Fraga).

1383-7621/00/$ - see front matter
PII: S1383-7621(00)00004-7

tasks" [14]. The uncertainty about the release time of the successor task can be modeled as a release jitter [18].

Most studies on precedence relations deal only with linear precedences ("pipelines"), where each task has at most a single predecessor and a single successor. Task models that include arbitrary precedence relations do not impose such restriction. Studies that consider arbitrary precedence relations usually assume statically released tasks instead of the dynamically released tasks. Also, only a few papers consider this problem in a distributed environment.

The schedulability of statically released tasks with arbitrary precedence relations executing on a single processor is analyzed in [3,8]. In [5] arbitrary precedence relations are considered in a distributed environment, also for statically released tasks. Linear precedence relations are analyzed in [7,9,12–16].

The work described in [2] implements arbitrary precedence relations by dynamically releasing tasks. It shows that release jitter can be used to model precedence relations caused by communication between tasks in different processors. That paper does not develop the approach in a detailed way.

The work presented in [6] develops an appropriate schedulability analysis for distributed systems built on a point-to-point network. The task model includes arbitrary precedence relations, fixed priority and dynamically released tasks. Precedence relations are transformed into release jitter, so all tasks can be analyzed as they were independent.

Similarly, [18] presents a schedulability analysis for distributed applications on a bus-based network. It assumes fixed priority and dynamically released tasks. Linear precedence relations are transformed into release jitter, so all tasks are analyzed as they were independent. Arbitrary precedence relations are possible through statically released tasks.

In this paper we consider the scheduling of hard real-time tasks with arbitrary precedence constraints in a distributed environment. We assume a task model that includes fixed priority and dynamic release of tasks. The objective is to present a technique for schedulability analysis that is valid for systems where tasks are released as soon as the necessary messages have arrived. We develop transformation rules that, starting from the original task set, create equivalent sets of independent tasks with release jitter. These new sets of independent tasks can be used to evaluate the schedulability of the original task set.

The method presented in this paper to transform arbitrary precedence relations into release jitter is less pessimistic than a simple direct transformation as presented in [6]. By eliminating all precedence relations present in the original task set one arrives at a set of independent tasks. This new task set can be analyzed by any available schedulability test that is valid for independent task sets.

The remainder of the paper is organized as follows: Section 2 describes the approach adopted in this work; Section 3 formulates the problem; in Section 4 we develop a set of transformation rules; Section 5 describes a schedulability test algorithm; simulation results are presented in Section 6; finally, concluding remarks are presented in Section 7.

2. Description of our approach

There are in the literature several tests to analyze the schedulability of independent task sets when fixed priorities are used. Some of those tests allow tasks with release jitter, such as [2,17]. Those two works in particular analyze the schedulability of each task by computing its maximum response time and then comparing this value with the deadline.

It is possible to make a simple and direct transformation of precedence relations to release jitter [6]. When a task T_i has many predecessor tasks it is enough to identify the predecessor task that will cause the maximum delay to the start of T_i. This delay includes the maximum response time of the predecessor task plus any communication delay due to inter-processor message passing. This maximum delay to the start of T_i is transformed in the release jitter of task T_i. Its precedence relations can now be ignored during the schedulability analysis [6]. The schedulability