Automatica 45 (2009) 2597–2604

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

Supervisory control for real-time scheduling of periodic and sporadic tasks with resource constraints[☆]

Seong-Jin Park[a], Jung-Min Yang[b,*]

[a] *Department of Electrical and Computer Engineering, Ajou University, Suwon, 443-749, Korea*

[b] *Department of Electrical Engineering, Catholic University of Daegu, Kyongsan Kyongbuk, 712-702, Korea*

ARTICLE INFO

Article history:
Received 25 August 2008
Received in revised form
21 May 2009
Accepted 7 July 2009
Available online 19 August 2009

Keywords:
Supervisory control
Timed discrete event systems
Real-time scheduling
Resource constraints
Deadlock

ABSTRACT

In the framework of supervisory control of timed discrete event systems, this paper addresses the design problem of a real-time scheduler that meets stringent time constraints of periodic tasks and sporadic tasks which exclusively access shared resources. For this purpose, we present the timed discrete event models of execution of periodic tasks and sporadic tasks and resource access for shared resources. Based on these models, we present the notion of deadlock-free and schedulable languages that contain only deadline-meeting sequences which do not reach deadlock states. In addition, we present the method of systematically computing the largest deadlock-free and schedulable language, and it is also shown that schedulability analysis can be done using this language. We further show that the real-time scheduler achieving the largest deadlock-free and schedulable language is optimal in the sense that there are no other schedulers to achieve schedulable cases more than those achieved by the optimal scheduler.

1. Introduction

A scheduler in a real-time system coordinates the execution of tasks in order to meet their given time constraints (deadlines). The theory of real-time scheduling has been mainly developed in the computer science community. Two fundamental scheduling problems can be summarized as follows: One is the schedulability analysis problem, i.e. is it possible to meet the deadlines of given tasks? The other is the scheduler design problem which is also called scheduling algorithm selection problem (Liu, 2000). When several tasks exclusively access shared resources, these scheduling problems become much more difficult because avoiding deadlocks should be considered together with meeting deadlines (Baruah, 2006; Fisher, Bertogna, & Baruah, 2007; Lipari & Buttazzo, 2000; Silly-Chetto, 1999).

Recently, several researchers have shown that the real-time schedulers can be effectively designed by synthesizing feedback controllers (Lu, Stankovic, Tao, & Son, 2002). In particular, it has been shown that the supervisory control theory of discrete event systems (DESs) can be successfully applied to the design of schedulers. In Altisen, Goessler, and Sifakis (2002), the scheduler design method based on timed automata and state feedback control was presented. The design method for the non-preemptive scheduling of periodic tasks with deadlines was reported in Chen and Wonham (2002), and the scheduler synthesis method for periodic tasks with fixed priorities was reported in Janarthanan, Gohari, and Saffar (2006). The scheduling problem for multiprocessors executing periodic tasks with fixed release times was also addressed in Janarthanan and Gohari (2007). In Onogi and Ushio (2006), the scheduling method for the non-preempting periodic tasks in dynamically reconfigurable devices was presented. In Park and Cho (2008), the optimal scheduler for sporadic tasks with arbitrary release times was developed. In Park and Cho (2009), the optimal scheduler for sporadic tasks executing on a multiprocessor was developed in the presence of processor faults, and in Yang and Park (2008), schedulability analysis for periodic and sporadic tasks was done. However, the previous studies have dealt with the scheduling problems without considering resource constraints.

In this paper, we address the scheduling problems of periodic tasks and sporadic tasks with time and resource constraints using the supervisory control theory of timed DESs. Periodic tasks are assumed to be invoked with fixed periods and have no need for their deadlines to be equivalent to their periods. Sporadic tasks are assumed to be invoked with arbitrary release times and rejected or accepted for processing. The deadlines of tasks are also assumed to be known as constant values a priori. Under these assumptions, we present the timed discrete event models for task execution and resource access, and formulate the schedulability analysis problem and the design problem of optimal schedulers. To solve these

[☆] The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor George Yin under the direction of Editor Ian R. Petersen.

[*] Corresponding author. Tel.: +82 53 850 2736; fax: +82 53 850 2704.
E-mail addresses: parksjin@ajou.ac.kr (S.-J. Park), jmyang@cu.ac.kr (J.-M. Yang).

0005-1098/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2009.07.011

problems, we introduce the notion of a deadlock-free and schedulable language composed of deadline-meeting sequences which do not reach deadlock states, and additionally provide the systematic method of computing the largest deadlock-free and schedulable language. We further show that the schedulability analysis problem can be solved using the language. An optimal scheduler is designed in an offline manner based on the largest deadlock-free and schedulable language, and the scheduler coordinates the execution and resource access of tasks in an online manner upon observation of the executed sequences. In addition, the scheduler decides whether each newly arrived sporadic task is accepted or rejected.

The scheduling problems presented in this paper are derived from the setting of Yang and Park (2008) which does not consider resource constraints. Specifically, the timed discrete event models presented for periodic and sporadic tasks are extended from those of Yang and Park (2008) through including the resource access of tasks. In addition, we establish the timed discrete event models for the resource access of each task and the exclusive access constraint to each resource which are not presented in Yang and Park (2008). Moreover, the scheduling policy presented in this paper is different from that of Yang and Park (2008). Even though the acceptance of a newly arrived sporadic task results in meeting the deadlines of tasks, the scheduler of Yang and Park (2008) may reject the arrived task. However, the scheduler presented in this paper always accepts newly arrived sporadic tasks if the acceptance of the tasks can lead to meeting their deadlines. In other words, the presented scheduler has a policy that it should process as many tasks as possible if it can meet all the deadlines of tasks and also avoid deadlocks.

The supervisory control approach presented in this paper for real-time scheduling differs from the conventional approaches by the computer science community in several aspects. While conventional approaches separately deal with the problems of finding the schedulability conditions (the solutions of a schedulability analysis problem) and realizing the specific scheduling algorithms such as EDF (Earliest Deadline First), the supervisory control approach simultaneously solves these problems by systematically computing a deadlock-free and schedulable language. It means that the presented approach is more concise and efficient in formulating and solving the scheduling problems. In addition, while the conventional scheduling solutions usually result in only partial execution sequences that meet deadlines and avoid deadlocks, the presented largest deadlock-free and schedulable language is always complete in that it contains all achievable execution sequences that meet deadlines and also avoid deadlocks.

2. Timed discrete event model (TDEM)

We present a new TDEM that adds the memorable events to the TDEM framework of Brandin and Wonham (1994). In a real-time system, when the execution of a task is preempted by another task, the interrupted task falls into a blocked state. Since its re-execution begins from the interrupted point generally, the task's timer containing the remaining time until its completion should be maintained in the blocked state. This is the reason for introducing the notion of memorable events in this paper.

A TDEM consists of two elements, an activity (untimed) model and time bounds of events. First, the activity model is described by a finite state automaton

$$G_{act} = (A, \Sigma_{act}, \delta_{act}, a_0, A_m),$$

where A is a finite set of activities, Σ_{act} is a finite set of events, $\delta_{act} : A \times \Sigma_{act} \to A$ is an activity transition function, $a_0 \in A$ is an initial activity, and $A_m \subseteq A$ is a set of marked activities. For $\sigma \in \Sigma_{act}$, lower and upper time bounds are given as $\text{lower}(\sigma) \in N$ and $\text{upper}(\sigma) \in N \cup \{\infty\}$, respectively, where N denotes the set of all nonnegative integers. Σ_{act} is partitioned into $\Sigma_{spe} = \{\sigma \in \Sigma_{act} \mid \text{upper}(\sigma) \in N\}$ and $\Sigma_{rem} = \{\sigma \in \Sigma_{act} \mid \text{upper}(\sigma) = \infty\}$. Σ_{spe} (respectively, Σ_{rem}) is the set of prospective (respectively, remote) events whose upper time bounds are finite (respectively, infinite). In addition, we introduce a set of memorable events, $\Sigma_{mem} \subseteq \Sigma_{act}$. For a memorable event not defined at a state, its timer value at the state is maintained as its timer value at a previous state. For a non-memorable event not defined at a state, its timer value at the state is reset to its lower time bound or upper time bound. For $\sigma \in \Sigma_{act}$, its timer interval T_σ is defined as follows: $T_\sigma = \{i \in N \mid 0 \le i \le \text{upper}(\sigma)\}$ if $\sigma \in \Sigma_{spe}$; $T_\sigma = \{i \in N \mid 0 \le i \le \text{lower}(\sigma)\}$ if $\sigma \in \Sigma_{rem}$.

A TDEM G is then represented by the following finite state automaton

$$G = (Q, \Sigma, q_0, \delta, Q_m).$$

The state set Q is defined as $Q = A \times \Pi\{T_\sigma \mid \sigma \in \Sigma_{act}\}$, and a state $q \in Q$ is of the form $q = (a, \{t_\sigma \mid \sigma \in \Sigma_{act}\})$ in which t_σ is a timer of an event σ. The initial state $q_0 \in Q$ is defined as $q_0 = (a_0, \{t_{\sigma,0} \mid \sigma \in \Sigma_{act}\})$ where $t_{\sigma,0} = \text{upper}(\sigma)$ for $\sigma \in \Sigma_{spe}$ and $t_{\sigma,0} = \text{lower}(\sigma)$ for $\sigma \in \Sigma_{rem}$. The marked state set $Q_m \subseteq Q$ is given by a subset of $Q_m = A_m \times \Pi\{T_\sigma \mid \sigma \in \Sigma_{act}\}$. The event set Σ is defined as $\Sigma = \Sigma_{act} \cup \{tick\}$, where *tick* denotes the passage of one unit time.

By extending the framework of Brandin and Wonham (1994) with the consideration of memorable events, we can define the transition function $\delta : Q \times \Sigma \to Q$ as follows. Let $\delta(q, \sigma) = q'$ with $q = (a, \{t_\tau \mid \tau \in \Sigma_{act}\})$ and $q' = (a', \{t'_\tau \mid \tau \in \Sigma_{act}\})$. Then $\delta(q, \sigma)$ is defined if and only if

(1) $\sigma = tick$ and $(\forall \tau \in \Sigma_{spe})\ t_\tau > 0$, or
(2) $\sigma \in \Sigma_{spe}$, $\delta_{act}(a, \sigma)!$, and $0 \le t_\sigma \le \text{upper}(\sigma) - \text{lower}(\sigma)$, or
(3) $\sigma \in \Sigma_{rem}$, $\delta_{act}(a, \sigma)!$, and $t_\sigma = 0$.

The symbol ! means 'is defined'. The entrance state q' is defined as follows.

(1) If $\sigma = tick$, then $a' := a$ and for each $\tau \in \Sigma_{act}$,

(a) if $\tau \in \Sigma_{spe}$,

$$t'_\tau := \begin{cases} u_\tau & \text{if } [\delta_{act}(a,\tau) \text{ is not defined}] \wedge [\tau \notin \Sigma_{mem}], \\ t_\tau & \text{if } [\delta_{act}(a,\tau) \text{ is not defined}] \wedge [\tau \in \Sigma_{mem}], \\ t_\tau - 1 & \text{if } [\delta_{act}(a,\tau)!] \wedge [t_\tau > 0]. \end{cases}$$

(b) if $\tau \in \Sigma_{rem}$,

$$t'_\tau := \begin{cases} l_\tau & \text{if } [\delta_{act}(a,\tau) \text{ is not defined}] \wedge [\tau \notin \Sigma_{mem}], \\ t_\tau & \text{if } [\delta_{act}(a,\tau) \text{ is not defined}] \wedge [\tau \in \Sigma_{mem}], \\ t_\tau - 1 & \text{if } [\delta_{act}(a,\tau)!] \wedge [t_\tau > 0], \\ 0 & \text{if } [\delta_{act}(a,\tau)!] \wedge [t_\tau = 0]. \end{cases}$$

(2) If $\sigma \in \Sigma_{act}$, then $a' := \delta_{act}(a, \sigma)$ and for any $\tau \in \Sigma_{act}$,

(a) if $\tau \ne \sigma$ and $\tau \in \Sigma_{spe}$,

$$t'_\tau := \begin{cases} u_\tau & \text{if } [\delta_{act}(a',\tau) \text{ is not defined}] \wedge [\tau \notin \Sigma_{mem}], \\ t_\tau & \text{if } [\delta_{act}(a',\tau) \text{ is not defined}] \wedge [\tau \in \Sigma_{mem}], \\ t_\tau & \text{if } \delta_{act}(a',\tau)!. \end{cases}$$

(b) if $\tau = \sigma$ and $\tau \in \Sigma_{spe}$, $t'_\tau := \text{upper}(\sigma)$.

(c) if $\tau \ne \sigma$ and $\tau \in \Sigma_{rem}$,

$$t'_\tau := \begin{cases} l_\tau & \text{if } [\delta_{act}(a',\tau) \text{ is not defined}] \wedge [\tau \notin \Sigma_{mem}], \\ t_\tau & \text{if } [\delta_{act}(a',\tau) \text{ is not defined}] \wedge [\tau \in \Sigma_{mem}], \\ t_\tau & \text{if } \delta_{act}(a',\tau)!. \end{cases}$$

(d) if $\tau = \sigma$ and $\tau \in \Sigma_{rem}$, $t'_\tau := \text{lower}(\sigma)$.

Consider a simple task of which the activity model is shown in Fig. 1(a) where the initial activity and marked activity are denoted by the entering arrow and double circles, respectively. The time