On Adaptive Hierarchical Scheduling of Real-time Systems Using a Feedback Controller

Nima Moghaddami Khalilzad, Moris Behnam, Thomas Nolte and Mikael Åsberg
MRTC/Mälardalen University
P.O. Box 883, SE-721 23 Västerås, Sweden
nmi09001@student.mdh.se

***Abstract*—Hierarchical scheduling provides predictable timing and temporal isolation; two properties desirable in real-time embedded systems. In hierarchically scheduled systems, subsystems should receive a sufficient amount of CPU resources in order to be able to guarantee timing constraints of its internal parts (tasks). In static systems, an exact amount of CPU resource can be allocated to a subsystem. However, in dynamic systems, where execution times of tasks vary considerably during run-time, it is desirable to give a dynamic portion of the CPU given the current load situation. In this paper we present a feedback control approach for adapting the amount of CPU resource that is allocated to subsystems during run-time such that each subsystem receives sufficient resources while keeping the number of deadline violations to a minimum. We also show an example simulation where the controller adapts the budget of a subsystem.**

I. Introduction

Embedded real-time systems become increasingly more complex, making it difficult to combine hard real-time guarantees with efficient use of system resources. When run-time behavior of tasks in a complex real-time system is difficult to predict, the feedback scheduling concept can be used as a powerful tool for adapting scheduling to the task's requirements. Scheduling parameters can be adapted during run-time such that tasks get a better service in response to their request for the shared resources. Although a variety of techniques are available based on feedback scheduling, a suitable technique should be designed given the context of our Hierarchical Scheduling Framework (HSF) [1].

The HSF provides a modular way for scheduling and guarantying timing constraints of real-time tasks [2], [3]. The HSF can be illustrated using a tree structure in which each node is responsible for scheduling its children using resources received from its corresponding parent node. Each child provides the parent with parameters such as period and budget (the subsystem interface), and parents schedule their children according to the subsystem interface parameters. Resource efficient interface variables can be found, for example, by assuming a fixed period for subsystems and trying to find a minimum possible value for the budget in which the system is schedulable [4]. In this paper, a feedback mechanism is introduced for online control of the interface parameters in a HSF. The goal of the presented approach is to adapt the budget of subsystems during run-time to achieve an efficient CPU utilization in comparison with systems having pre-assigned fixed interface parameters, especially when tasks within a subsystem experience a considerable change in their execution time. Given a particular subsystem period, the subsystem budget should be kept to a minimum while at the same time minimizing the number of potential deadline misses within a predetermined time-interval. The contributions of this paper are the design of the feedback control system for dynamic adaptation of resource parameters in the HSF, and a simulation study investigating the performance of our solution.

The remainder of this paper is organized as follows. Section II provides a brief information about our HSF. Section III describes the mathematical model of the plant as well as the designed controller. Section IV shows one example simulation of our HSF with feedback control. Related works are presented in Section V. Finally, conclusion is presented in Section VI.

II. The Hierarchical Scheduling Framework

In this paper we investigate feedback scheduling in a single CPU where each CPU is modeled as a system S. Each system consists of a set of subsystems $S_S \in S$. The system is scheduled using a two level HSF. During run-time, the global scheduler chooses one of the subsystems and allocates CPU to that subsystem. Then, the subsystem's local scheduler shares this allocated CPU among its tasks according to its scheduling algorithm.

A. Subsystem Model

Each subsystem S_S is represented by its timing interface parameters (P_S, B_S) where P_S and B_S are subsystem period and budget respectively. Each subsystem S_S also consists of a set of tasks τ_S and a local scheduler.

B. Task Model

We assume the periodic soft real-time task model $\tau_i(T_i, P_i, C_i, D_i)$, where T_i, P_i, C_i and D_i are task period, priority, worst-case execution time and relative deadline respectively.

III. Controller

The objective of this section is to provide detailed information about how control theory is applied to our HSF. The controller changes the *manipulated variables* based on the input, which is the controller error, and the controller algorithm. The controller error is defined as the difference between *controlled variable* and the *reference input*. The first step in designing a controller is to define the controlled and

manipulated variables, which are explained in this section. In the design of the controller we have used a similar approach as the one presented in [5]. The significance of our work is that we apply feedback control to the context of hierarchical scheduling. We use two feedback loops to control the plant. The first loop is responsible for controlling the number of deadline misses and the second loop tries to reduce amount of idle time in the subsystems. To simplify the analyses, these feedback loops are considered to be independent from each other. Therefore, for each loop a set of controlled variables as well as analyses are presented seperately. It is important to notice that in both design and simulation phases we assume there are enough resources for all subsystems such that increasing budget of one subsystem does not affect timing guarantees of other subsystems.

A. Controlled Variables

The first controlled variable is $M_S(t)$ which is defined as the total number of missed deadline jobs of all tasks inside the subsystem $\tau_i \in S_S$, within one specified time window (tw_m) prior to the current time t. The control loop which uses $M_S(t)$ as its controlled variable is called "M-loop" in the rest of the paper. The next controlled variable is $U_S(t)$ that is defined by the following formula

$$U_S(t) = \frac{B_S(t)}{E_S(t)}$$

where $B_S(t)$ and $E_S(t)$ represent total budget of the subsystem and total measured time of CPU usage by all tasks of the subsystem S_S in the time window tw_u respectively. Similar to the M-loop, we call the second control loop "U-loop" in the rest of the paper.

B. Manipulated Variables

The budget of subsystem $B_S(t)$ is considered as the manipulated variable. The budget should be adjusted based on the error between the controlled variable and the reference input. In each sampling period, the controller adds budget change value $D_{B_S}(t)$ to the previous value of the subsystem budget.

$$B_S(t) = B_S(t-1) + D_{B_S}(t)$$

C. Model of Plant

In this section an approximate analytical model of the controlled system is presented. This model is useful when we are looking for optimal values of the tunable parameters in the controller. Based on the model and some analyses we find boundaries for tunable variables. Relation between the control output ($D_{B_S}(t)$) and controlled variables ($U_S(t)$ and $M_S(t)$) is of interest in the plant model. For the controlled variable $U_S(t)$, from the definition we have

$$U_S(t) = \frac{B_S(t)}{E_S(t)}.$$

In order to continue analysis we use $WCET_S = max(E_S(t))$. Hence

$$U_S(t) = \frac{B_S(t)}{WCET_S}$$

where $WCET_S$ is a total worst case execution time of all tasks in subsystem S_S. After transfering to the z domain, from the control input $D_B(z)$ to $U(z)$ the transfer function is: $U(z) = P_U(z)D_B(z)$ and $P_U(z) = G_U/(z-1)$ where $G_U = \frac{1}{WCET_S}$.

We can define $M_{S_S}(t)$ based on U_{S_S} and derive a similar model for $M_{S_S}(t)$:

$$M_{S_S}(t) = M_{S_S}(t-1) + G_m(U_{S_S}(t) - U_{S_S}(t-1))$$

where G_m is deadline miss factor and can be found by plotting the $M_{S_S}(t)$ curve as a function of $U_{S_S}(t)$. For continuing the analysis we use G_M as the maximum value of G_m. Similar to $U_{S_S}(t)$ we can derive the transfer function $P_M(z) = G_U G_M/(z-1)$.

D. Model of Controller

A PI controller is used to control the plant. As it is mentioned in [5] the rationale behind not using the derivative term (D) is that this term might amplify noise when system load experiences significant changes. The PI controller function is

$$D_{B_S}(t) = K_P Erorr_S(t) + K_I \sum_{tw} Error_S(t)$$

where K_P, K_I, $Erorr_S(t)$ and tw are proportional gain, integral gain, error value of the subsystem S_S at time t and time window respectively. Each control loop has its own controller. Therefore, introduced parameters are specific to each loop. These variables are tunable parameters of the controller and should be tuned to get a desirable performance. After applying the z-transform we have

$$D_{B_S}(z) = K_P + \frac{K_I}{(z-1)}.$$

E. Closed-Loop System Model

If we consider $G = G_U G_M$ for the M-loop and $G = G_U$ for the U-loop, we can derive the following closed-loop system model for both loops:

$$H_S(z) = \frac{C(z)P(z)}{1 + C(z)P(z)} = \frac{GK_P(z-1) + K_I G}{(z-1)^2 + G(K_P(z-1) + K_I)} \quad (1)$$

F. Stability Analysis

From (1) the characteristic equation is:

$$(z-1)^2 + G(K_P(z-1) + K_I) = z^2 + \alpha_1 z + \alpha_2$$

where $\alpha_1 = GK_P - 2$ and $\alpha_2 = 1 - GK_P + GK_I$. According to Jury's scheme [6, p. 82] the stability conditions are:

$$\alpha_2 < 1,\ \alpha_2 > -1 + \alpha_1 \text{ and } \alpha_2 > -1 - \alpha_1.$$

These conditions give us boundaries on tunable variables of the system.

IV. Simulation Results

The simulation environment is prepared by modeling the HSF in the Times[1] tool and generating a C++ file from the model [8]. The generated code is extended so that it

[1]Times is a tool for modeling and implementation of embedded systems [7]. Since Times supports *task automata* (*timed automata with tasks*), it is used for modeling, verification and code synthesis purposes.

contains the designed PI controller function. In addition, some functions are added for calculating the controlled variables in the scheduler body. Among all simulations conducted on the prepared scheduler, due to the space limitation in this paper, we present an example scenario that shows budget adaptation in situations which the execution time of a task in one subsystem varies from low to high and vice versa. We show how the budget is changed in response to the new load condition of the subsystem.

There are totally two subsystems in the system. In both global and local levels we use the fixed priority algorithm for scheduling subsystems and tasks. When a deadline miss happens, the task continues executing until it finishes. Specifications of subsystems are shown in Table I. We assume that tasks which are inside S_1 have a fixed execution time and that using the pre-assigned budget they can meet their deadlines. In subsystem S_2 there are two tasks, and their specifications are shown in Table II. We also assume that task one ($\tau 1$) experiences some changes in its execution time during run-time. Execution time changes are shown in Table III. The execution time variation is done using a function which is responsible for changing execution time of tasks to a predefined value at a specific clock cycle. In the presented simulation, execution time is changed in a range such that it does not violate the whole system schedulability condition.

Name	P_S	B_S	Priority
S_1	19	2	1
S_2	5	3	0

TABLE I
SUBSYSTEMS SPECIFICATIONS

Name	T_i	D_i	P_i	C_i
$\tau 1$	10	6	1	3
$\tau 2$	11	8	0	1

TABLE II
TASKS SPECIFICATIONS OF S_1

Time	0	50	200	400
C_i	3	2	3	0

TABLE III
EXECUTION TIME CHANGES OF $\tau 1$

The *tw* and controller period are considered to be 15 in this example. Hence, every 15 ticks the controller measures the controlled variables, and based on their value takes action by changing the budget of the subsystem one S_1. The controller period is experimentally tuned by taking into consideration the trade-off between calculation overhead and the controller response speed. In order to have a faster reaction to the environment changes, we can decrease the controller period. Consequently, it can sample and actuate more frequently which however increases the run-time overhead. After changing the controller period, tunable variables of the controller should be tuned to acquire a better controller performance. The controller is implemented inside the scheduler such that the scheduler runs the controller function (periodically) before other parts of the code.

The system is executed for 600 ticks, and the controlled variables as well as the budget are sampled in each controller execution, and the result is illustrated in Figure 1. As it is shown in Figure 1, 15 ticks after the system starts execution, the controller observes one deadline miss. It means that the pre-assigned budget is not enough for the tasks of S_1 to meet their deadlines. After observing the deadline miss, the controller increases the budget and after that all subsystem tasks are able to finish execution before their corresponding deadline. At time 50, when the execution time of $\tau 1$ is reduced from three to two, the controller reduces the budget from four to two in two steps. At time 200, the execution time of $\tau 1$ is increased to three and it causes some deadline misses. After which the controller observes the deadline misses, it increases the budget. Finally, the last change happens at time 400, when the execution time of $\tau 1$ is reduced to zero. In this case the U-loop experiences a huge error value which is conducive to a sudden change of the budget from four to three and eventually to one.

The important point to highlight here is that when we move from low to high execution time, the M-loop plays an essential role in adapting the budget. On the other hand, when the execution time is decreased, the U-loop adapts the budget according to the current requirements of the system.

In order to illustrate the difference between having an adaptive budget and having a pre-assigned fixed budget, we have conducted a set of simulations using a fixed budget and we have measured the amount of idle time and the number of deadline misses of S_1 during first 600 ticks. Table IV shows a comparison between using different budgets and using our adaptive approach. Since the scheduler does not support execution of a task after its period, we couldn't measure values for the budget equal to one.

Budget	3	2	1	adaptive
Deadline misses	12	33	-	4
Idle time	197	77	-	185

TABLE IV
IDLE TIME AND DEADLINE MISSES USING DIFFERENT BUDGETS

V. RELATED WORKS

Feedback scheduling has been used in scheduling of control tasks for acquiring predictable performance when execution time of tasks are subjected to sudden changes [9]. Model Predictive Controllers (MPC) are scheduled using a feedback loop [10]. In [11] feedback-based scheduling is used in the real-time memory garbage collector.

Feedback scheduling applied to reservation-based algorithms and a complete mathematical analysis is presented in [12]. In [13] a two level controller is proposed to share resources among a pipeline of tasks and satisfy Quality of Service (QoS) requirements. Scheduling of tasks was investigated in the stochastic domain and a two block controller was suggested [14]. In [15] a two-level feedback controller in the context of a reservation technique is introduced, where application level and system level QoS are improved based on bandwidth adaptation.

In [16] optimizing techniques are used for controlling the CPU utilization in multiprocessor systems. Stankovic et

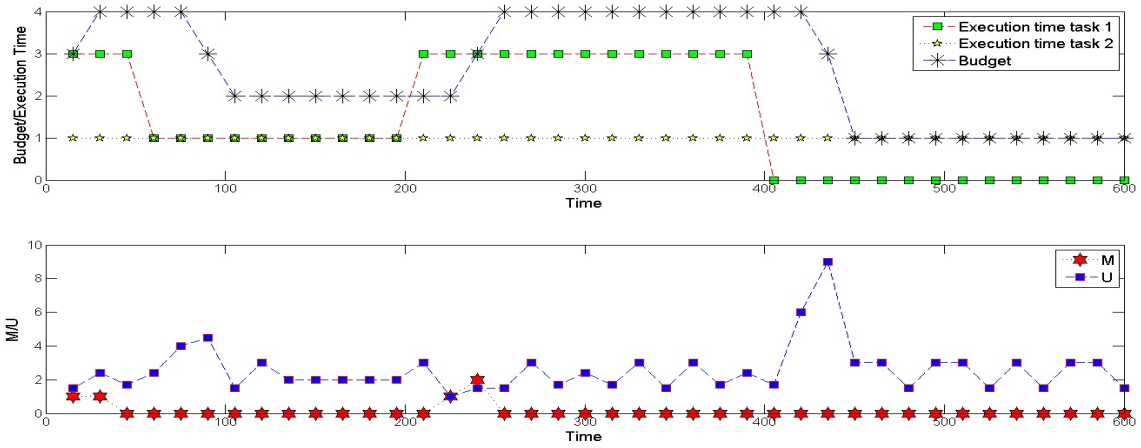

Fig. 1. Execution times, budget and controlled variables change over time

al. have applied feedback control techniques in distributed systems [17] and they have proposed local and global level feedback controllers. Lu et al. introduced a Proportional Integral Derivative (PID) controller which controls CPU utilization requests based on miss ratio feedback [18]. They continued their work and presented a two-feedback loop system [5]. None of the aforementioned techniques have been applied in the context of HSF.

VI. Summary and Conclusions

In this paper we have used feedback control techniques in the context of a hierarchical scheduling framework for adapting the budgets of subsystems during run-time. Simulation results show that the controller is able to adapt the budget when execution times of tasks are changed. When the system is not overloaded, manipulating the budget of one subsystem either lets more subsystem tasks to meet their deadlines or it decreases the response time of the tasks that are inside the other subsystems in the same node.

Our next step is to investigate multi-mode realtime systems [19] and integrate hierarchical scheduling with the mode shift concept. The budget value in different modes can be adapted using the introduced feedback loops. In addition, we will look into situations when by changing the budget of a subsystem, the system becomes not schedulable. In this paper we have investigated budget change in a single subsystem; however, in the future we will study more complicated systems in which two or more subsystems are subjected to the budget change. Besides, we will conduct some experiments on a real system to study performance of our adaptive framework. Finally, investigating and applying feedback techniques in multicore HSFs is another trend of our work.

References

[1] T. Nolte, M. Behnam, M. Åsberg, R. J. Bril, and I. Shin, "Hierarchical scheduling of complex embedded real-time systems," in *Ecole d'Ete Temps-Reel (ETR'09)*, August 2009, pp. 129–142.

[2] Z. Deng and J. W.-S. Liu, "Scheduling real-time applications in an open environment," in *Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS '97)*, 1997, pp. 308–.

[3] G. Lipari and S. Baruah, "A hierarchical extension to the constant bandwidth server framework," in *Proceedings of the Seventh IEEE Real-Time Technology and Applications Symposium*, 2001, pp. 26 –35.

[4] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proceedings of the 24th IEEE Real-Time Systems Symposium, RTSS*, 2003, pp. 2 – 13.

[5] C. Lu, J. A. Stankovic, S. H. Son, and G. Tao, "Feedback control real-time scheduling: Framework, modeling, and algorithms*," *Real-Time Systems*, vol. 23, pp. 85–126, 2002.

[6] B. W. Karl Johan Astrom, *Computer-Controlled Systems: Theory and Design (3rd Edition)*. Prentice Hall, 1996.

[7] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, "Times - a tool for modelling and implementation of embedded systems," in *Proceedings of 8th International Conference, TACAS 2002, (ETAPS 2002)*, April 2002, pp. 460–464.

[8] M. Åsberg, P. Pettersson, and T. Nolte, "Modelling, verification and synthesis of two-tier hierarchical fixed-priority preemptive scheduling," Mälardalen University, Technical Report, March 2011.

[9] A. Cervin and J. Eker, "Feedback scheduling of control tasks," in *Proceedings of the 39th IEEE Conference on Decision and Control*, 2000, pp. 4871 –4876 vol.5.

[10] D. Henriksson, A. Cervin, J. Akesson, and K.-E. Arzen, "Feedback scheduling of model predictive controllers," in *Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium*, 2002, pp. 207 – 216.

[11] S. Robertz, D. Henriksson, and A. Cervin, "Memory-aware feedback scheduling of control tasks," in *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA '06)*, 2006, pp. 70 –77.

[12] L. Abeni, L. Palopoli, G. Lipari, and J. Walpole, "Analysis of a reservation-based feedback scheduler," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS '02)*, 2002, pp. 71–.

[13] T. Cucinotta and L. Palopoli, "Feedback scheduling for pipelines of tasks," in *Proceedings of the 10th international conference on Hybrid systems: computation and control (HSCC'07)*, 2007, pp. 131–144.

[14] T. Cucinotta, L. Palopoli, and L. Marzario, "Stochastic feedback-based control of qos in soft real-time systems," in *Proceedings of the 43rd IEEE Conference on Decision and Control*, 2004, pp. 3533 – 3538 Vol.4.

[15] L. Abeni and G. Buttazzo, "Hierarchical qos management for time sensitive applications," in *Proceedings of the Seventh Real-Time Technology and Applications Symposium (RTAS '01)*, 2001, pp. 63–.

[16] J. Yao, X. Liu, Z. Gu, X. Wang, and J. Li, "Online adaptive utilization control for real-time embedded multiprocessor systems," *Journal of Systems Architecture*, vol. 56, no. 9, pp. 463 – 473, 2010.

[17] J. A. Stankovic, T. He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu, "Feedback control scheduling in distributed real-time systems," in *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS '01)*, 2001, pp. 59–.

[18] C. Lu, J. Stankovic, G. Tao, and S. Son, "Design and evaluation of a feedback control edf scheduling algorithm," in *Proceedings of the 20th IEEE Real-Time Systems Symposium*, 1999, pp. 56 –67.

[19] L. T. X. Phan, I. Lee, and O. Sokolsky, "Compositional analysis of multi-mode systems," in *Proceedings of the 22nd Euromicro Conference on Real-Time Systems (ECRTS '10)*, 2010, pp. 197–206.