Mixed-criticality Hierarchical Scheduling for TI-RTOS

Hesham Hussien
Department of Computer System
FCIS, Ain Shams University
Cairo, Egypt
hesham.hussien@cis.asu.edu.eg

Eman Shaaban
Department of Computer System
FCIS, Ain Shams University
Cairo, Egypt
eman.shaaban@cis.asu.edu.eg

Said Ghonaimy
Department of Computer System
FCIS, Ain Shams University
Cairo, Egypt
ghoniemy1@cis.asu.edu.eg

***Abstract*—The paper proposes an adaptive hierarchical scheduling framework for a set of independent concurrent applications, which composed of set of tasks with different criticality. It ensures temporal partitioning between independent applications with budget adaption feature, where CPU time of each application is periodically and dynamically assigned. It implemented in kernel of TI-RTOS on a resource constrained platform, experiments show that it enhances the performance of our previously proposed AHSF-EDF during overload conditions, and ensures that critical tasks assigned to a server meet their deadlines at the expense of other tasks assigned to the same server.**

Keywords—Mixed-Criticality Systems (MCs); Adaptive Hierarchical Scheduling Framework (AHSF); Real-Time Systems; Resource Sharing.

I. INTRODUCTION

Recently, the embedded systems become more complicated, particularly when integrate diverse applications with different criticality levels. These types of complicated systems are called Mixed-Criticality systems (MCs) [2]. MC is classifying the criticality of the system functions so that ensuring protecting critical ones against failures, which has gained a lot of interest and researchers' contributions as in [1], [2], [5], [6], [8] and [17].

MCs must provide isolated and guaranteed required resources for the critical functions. Examples of MCs in the industry that integrate a group of functionalities in a single platform such as AUTOSAR (AUTomotive Open System ARchitecture) [10] for the automotive and IMA (Integrated Modular Avionics) [15] for aerospace.

In our previous work [11], we proposed **AHSF-EDF** (an Adaptive Hierarchical Scheduling Framework based on EDF) as a scheduling framework (global and local schedulers) for TI-RTOS, where the system consists of a group of applications which composed of hard (high-critical) and soft (low-critical) real-time tasks that run on a single processor. However the previous proposed local scheduler did not ensure that the critical tasks meet their deadlines during overload conditions.

To resolve this problem, we will replace the previous local scheduler in AHSF-EDF by an advanced scheduler [8] (local scheduler is our target in this work). Where different relative deadlines are assigned to tasks depending on their criticality modes. We consider dual-criticality levels (low-criticality LO and high-criticality HI).

In this paper, we propose an enhanced scheduling framework AHSF-VD (an Adaptive Hierarchical Scheduling Framework based on EDF with Virtual Deadline) for TI-RTOS, similar to our previous framework architecture in [11] with replacing its local scheduler. Where the virtual relative deadlines for high tasks are generated by greedy tuning algorithm described in [8].

The proposed AHSF-VD framework has the following strengths:

It supports hierarchical scheduling, which ensures temporal partitioning between independent applications with budget adaption feature that dynamically adjusts CPU time of each application.

The system designer does not need to provide execution time for tasks. AHSF-VD applies Chebyshev's predictor [3] to predict different execution times.

The system designer can specify which tasks are more important in mixed critical systems during overload conditions.

Implemented in kernel of TI-RTOS on a resource constrained platform, experiments show that proposed scheme provides good performance for multiple applications with dynamic tasks under overload conditions.

The rest of the paper is organized as follows. Section II reviews the related work. Section III reveals the details of proposed AHSF-VD. An illustrated example presented in Section IV. Section V shows the evaluation of the proposed framework. Section VI concludes the paper.

II. RELATED WORK

Scheduling of Mixed-criticality systems (MCs) was initially introduced by Vestal [2], which updates the worst-case response-time algorithm previously proposed by Joseph-Pandya [14] to analyze mixed-criticality systems that run on single-processor platforms. Posteriorly, an improvement has presented in response time analysis for fixed-priority scheduling by Baruah et al. [7].

Baruah [16] introduced designing scheduling algorithms that balance between the ability to make critical-tasks meet their deadlines and achieving high utilization of the platform resources. Another approach presented also by Baruah [17] which focused on directed acyclic graph (DAG) [4] to synchronize the tasks' dependencies in MCs.

Authors in [8] formulated the tasks' demand bound functions dbf in [13], and presented an efficient tuning algorithm to update relative deadline of critical tasks to shift task's execution demands from high to low criticality mode.

A Bailout protocol was proposed in [12]. It guaranteed demanded execution times for high-critical tasks, as well as minimizes the negative impact on low-critical tasks, via a timely return to normal operation. It has three modes: normal, bailout and recovery modes, where the normal mode is corresponding to LO-criticality mode while the others are corresponding to HI-criticality mode. Enhancements are applied to bailout protocol in [6], to reduce the time slots that assigned to bailout mode and that consumed in that mode.

In [9] a survey of diverse scheduling techniques and its analysis for MCs on single-processor /multi-processors is presented.

III. SYSTEM MODEL

This section describes the overall architecture of the proposed mixed-criticality hierarchical scheduling framework (AHSF-VD) shown in Fig. 1. The proposed approach assumes a system consists of a set of concurrent independent applications runs on a single-core processor. Each application runs on a server combines high-critical and/or low-critical real-time periodic tasks.

Each task is served based on the policies of local and schedulers decisions (described in section 3). Where the global scheduler selects which server to run, and the local scheduler selects which task to serve within a server.

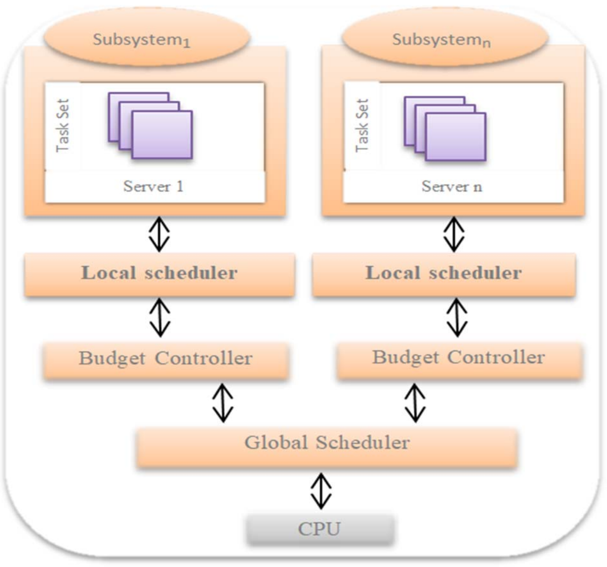

Fig. 1. Adaptive Hierarchical Scheduling Framework [11]

A. Server Model

Each server in AHSF-VD can be represented by several parameters (T_S, B_S, D_S, ξ_S, and M_S). Where T_S, B_S, D_S, ξ_S and M_S are server period, CPU budget, relative deadline, criticality level, and mode respectively. Detailed of server's budget assigning and adaptation algorithms are listed in [11]. Server mode Ms describes the current server's mode $M_S \in \{\mathbf{LO}, \mathbf{HI}\}$, where all servers start in low-mode M_S = LO. If a running server's task $t_{i,s}$ executes for its lower bound of execution time $EL_{i,s}$ and still requires extra time, then the scheduler upgrades the running server's mode to high-mode M_S = HI. On this mode the server serves only its high-critical tasks and excludes its low-critical tasks until all critical jobs are accomplished. Then the scheduler degrades the running server's mode to low-mode, and returns to serve all tasks.

B. Task Model

Each server is composed of a set of periodic low and high criticality real-time tasks. Each task is represented by several parameters ($T_{i,s}$, $\xi_{i,s}$, $EL_{i,s}$, $EH_{i,s}$, $D^L_{i,s}$, $D^H_{i,s}$), where $T_{i,s}$, $\xi_{i,s}$, $EL_{i,s}$, $EH_{i,s}$, $D^L_{i,s}$ and $D^H_{i,s}$ are period, criticality, lower and upper bounds on execution time, and relative low and high deadlines of task i in server S respectively. Values of $EL_{i,s}$ and $EH_{i,s}$ are estimated with Chebyshev's Inequality predictor [3]. Initial values of $D^L_{i,s}$ and $D^H_{i,s}$ are set to $T_{i,s}$.

Task criticality $\xi_{i,s} \in \{LO, HI\}$, where lower bound on execution time $EL_{i,s}$ is assigned to low-critical tasks, and high-critical tasks may be assigned lower and upper bounds on execution time $EL_{i,s}$ and $EH_{i,s}$. The task t_i in server S is subject to the constraint:

$$EL_{i,s} \leq EH_{i,s} \leq D^{L}_{i,s} \leq D^{H}_{i,s} \leq T_{i,s}$$

C. *Global scheduling*

The proposed framework AHSF-VD employs the same global scheduling policy implemented in AHSF-EDF, where global scheduling decision is based on server's deadline and critical level, and selects the server with the minimum deadline [11].

D. *Local Scheduling*

The local scheduler implements an efficient greedy algorithm based on EDF for tuning the relative low deadline of critical tasks $D^{L}_{i,s}$ described in [8].

In each server, its tasks can be scheduled by the local scheduler, if the schedulability test is passed for LO and HI critical modes:

$$U_S = \frac{B_s}{T_s} \tag{1}$$

$$U_S^{LO} = \sum_{i=0}^{n-1} \frac{EL_i}{T_i} \leq U_S \tag{2}$$

$$U_S^{HI} = \sum_{i=0}^{n-1} \frac{EH_i}{T_i} \leq U_S \tag{3}$$

Where B_s, T_s, U_s, U_s^{LO} and U_s^{HI} are budget, period, utilization, low utilization and high utilization of server s respectively, n is number of tasks assigned to server s.

The tuning algorithm picks candidate high-critical task set to be adjusted for its relative low deadline according to the following conditions:

1. Total low demand bound function dbf_{LO} for all tasks in server Si when running in low-mode during window interval L, does not exceed supply bound function sbf_{LO} (S_i, L).

2. Total upper demand bound function dbf_{HI} for critical tasks in server S_i when running in high-mode during window interval L, does not exceed supply bound function sbf_{HI} (S_i, L).

$$sbf_{LO}(S_i, L) = sbf_{HI}(S_i, L) = U_s * L \tag{4}$$

$$\forall L \geq 0: \sum_{ti \in t} dbf_{LO}(t_i, S_i, L) \leq sbf_{LO}(S_i, L) \tag{5}$$

$$\forall L \geq 0: \sum_{ti \in HI(t)} dbf_{HI}(t_i, S_i, L) \leq sbf_{HI}(S_i, L) \tag{6}$$

dbf_{LO} and dbf_{HI} are formulated in [8]. AHSF-VD assumes that window size L is equal to highest period of tasks in the system. The local scheduler makes its decisions based on the new tuning low-deadlines, where it chooses the task with the lowest relative deadline D^{L}_{i} while the server in low-mode. However if the server switched to high-mode, the scheduler defers the low-critical tasks and picks only among high-critical tasks, where it selects the task with the shortest deadline D^{H}_{i}.

The local scheduling decision is based on the task's deadline which selects the task with the shortest deadline, however if multiple tasks have the same deadline, it schedules the tasks based on FIFO mechanism.

IV. ILLUSTRATED EXAMPLE

The following example illustrates the local scheduling of task set shown in Table I in a dedicated system.

TABLE I. TASK SET FOR DUAL-CRITICALITY SYSTEM

Tasks	ξ_i	EL_i	EH_i	$T_i = D^{H}_{i} = D^{L}_{i}$
t1	LO	2	2	6
t2	HI	2	5	8
t3	HI	2	3	8

The scheduler starts with checking the schedulability for the task set as follows:

$$U_s = 1$$

$$U_S^{LO} = \frac{2}{6} + \frac{2}{8} + \frac{2}{8} = 0.833 \leq U_s$$

$$U_S^{HI} = \frac{5}{8} + \frac{3}{8} = 1 \leq U_s$$

TABLE II. UPDATES ON RELATIVE LOW DEADLINES FOR T2 & T3 AND L = 8.

iteration	0	1	2	3	4	5	6	7	8	9
L	0	0	1	1	2	2	3	4	5	8
D2	7	7	6	6	5	5	4	3	2	2
D3	8	7	7	6	6	5	5	5	5	5

Table II shows the results of applying greedy tuning algorithm on the task set. Relative deadline for critical tasks t2 and t3 is initially set to task period, where $D^H_i = D^L_i = T_i$. Updates on relative low deadline for high-critical tasks t2 and t3 with each iteration for window size L = 8 are shown.

V. Evaluation

This section evaluates the performance of the proposed scheduling framework AHSF-VD. Two different experiments are carried out to evaluate the performance of the enhanced framework, and compare it with our previously proposed AHSF-EDF in [11]. In each experiment, we run two different applications with three tasks each. The First application is a high-critical level application which runs on server S0 with tasks T0, T1 and T2, while the second application is a low critical level application runs on server S1 with tasks T3, T4 and T5. Table III & V show period of each task/server, task's criticality/server's mode for experiment I & II respectively.

A. Evaluation setup

We implemented AHSF-EDF and its enhanced version AHSF-VD into the kernel of TI-RTOS using TivaTM C Series TM4c123G Evaluation kit. 80 MHz 32-bit ARM CortexTM-M4 based MCU with floating point. 256KB flash, 32KB SRAM and 2KB EEPROM on-chip memory.

The Server window is set to 4, and kL and kH are set to 1.0 and 5.0 respectively.

- *Experiment I*

In the first experiment, we consider only periodic static tasks, with a slight variation between low and high bound execution time EL_i and EH_i for each task. Table III shows a snapshot of budgets and execution times for an overloaded situation after number of executions specified in Table IV. Deadline miss ratios for tasks of experiment I are listed in Table IV, where all tasks of high-critical server S0 that scheduled by AHSF-VD and AHSF-EDF meet their deadlines with no miss ratios.

Both AHSF-VD and AHSF-EDF ensure that all tasks of high-critical server S0 meet their deadlines with no miss ratios. For overloaded low-critical server S1, we observed that low-critical tasks (T3 and T4) miss ratios for AHSF-EDF are approximately three times miss ratios associated with AHSF-VD. Also, AHSF-VD proves a significant improvement in the scheduling of high-critical task T5, which meets its deadline with no miss ratio due to updating relative deadline D^L_3, compared to AHSF-EDF which reached to 35%.

Table III. Specification of servers and tasks for experiment I

S/T	**T**	**W/C**	**Budgets / Execution Time**					
			AHSF-EDF			***AHSF-VD***		
			ELi	***EHi***	***Bs***	***ELi***	***EHi***	***Bs***
S0	100	H	43	46	43	45	46	45
T0	200	L	23	25		24	24	
T1	300	H	46	50		47	51	
T2	200	L	32	33		33	33	
S1	100	L	59	62	57	60	61	55
T3	300	L	35	37		36	36	
T4	200	L	43	45		43	43	
T5	300	H	77	80		78	81	

Table IV. Performance of servers and tasks for experiment I

	Deadline Miss (%)		**No. of Executions**
	AHSF-EDF	***AHSF-VD***	
T_0	0.0%	0.0%	~3000
T_1	0.0%	0.0%	~2000
T_2	0.0%	0.0%	~3000
T_3	45%	15%	~2000
T_4	29%	9%	~3000
T_5	35%	0.0%	~2000

- *Experiment II*

In this experiment, we consider a set of static and dynamic tasks, where T0 and T2 are periodic static tasks, while T1, T3, T4 and T5 are periodic dynamic tasks. Table V shows a snapshot of budgets and execution times for an overloaded situation after number of executions specified in Table VI. Deadline miss ratios for tasks of experiment II are listed in Table VI.

Both AHSF-VD and AHSF-EDF frameworks ensure that all tasks of high-critical server S0 meet their deadlines with no miss ratios. However, for overloaded low-critical server S1, deadline miss ratio for low-critical tasks (T3 and T4) scheduled by AHSF-VD is high, and approximately two times that of AHSF-EDF.

Although high-critical task T5 misses its deadline with AHSF-EDF scheduling, AHSF-VD ensures that T5 meets its deadline at the expense of other low-critical tasks T3 and T4.

Table V. Specification of servers and tasks for experiment II

S/T	T	W/C	Budgets / Execution Time					
			AHSF-EDF			*AHSF-VD*		
			EL_i	EH_i	B_s	EL_i	EH_i	B_s
S0	100	H	42	51	42	42	50	42
T0	200	L	27	27		27	27	
T1	300	H	41	68		42	67	
T2	200	L	28	29		28	28	
S1	100	L	75	122	58	75	90	58
T3	300	L	67	96		69	69	
T4	200	L	63	109		63	63	
T5	300	H	61	105		61	104	

Table VI. Performance of servers and tasks for the experiment II

	Deadline Miss (%)		No. of Executions
	AHSF-EDF	*AHSF-VD*	
T_0	0.0%	0.0%	~3000
T_1	0.0%	0.0%	~2000
T_2	0.0%	0.0%	~3000
T_3	10%	18%	~2000
T_4	8%	16%	~3000
T_5	6%	0.0%	~2000

Table VII. CPU and Memory Costs

	AHSF-EDF		AHSF -VD	
CPU	*Experiment I*	*Experiment II*	*Experiment I*	*Experiment II*
	99%	98.4%	98.9%	98%
FLASH	18%		19%	
SRAM	42%		~42%	

Table VII shows CPU and Memory costs for both AHSF-EDF and AHSF-VD for both experiments I & II. However, we observed that CPU utilization for AHSF-EDF is slightly higher than that of AHSF-VD, and FLASH memory consumption for AHSF-VD is increased by 1% due to overhead associated with its tuning algorithm. However no observable extra SRAM is consumed in AHSF-VD compared to AHSF-EDF.

VI. CONCLUSION

The paper proposes an enhanced framework **AHSF-VD** (an Adaptive Hierarchical Scheduling Framework based on EDF with Virtual Deadline), that dynamically adjusts CPU budget of each server. It is implemented into the kernel of TI-RTOS, and tested in a real platform. It guarantees the minimum budgets for high-critical servers during overload periods, and also ensures that high-critical tasks meet their deadlines with no miss ratios at the expense of low-critical tasks. The paper presents a comparison with AHSF-EDF previously proposed and implemented in TI-RTOS.

REFERENCES

[1] S. Baruah, H. Li, and L. Stougie. Towards the design of certifiable mixed- criticality systems. In Real-Time and Embedded Technology and Applica- tions Symposium (RTAS), 2010 16th IEEE, pages 13 –22, april 2010.

[2] Vestal, S. (2007, December). Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International (pp. 239-243). IEEE.

[3] Liu M, Behnam M, Kato S, Nolte T. An adaptive server-based scheduling framework with capacity reclaiming and borrowing. In Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on 2014 Aug 20 (pp. 1-10). IEEE.

[4] Jensen, F. V. (1996). An introduction to Bayesian networks (Vol. 210, pp. 1-178). London: UCL press.

[5] Guan, N., Ekberg, P., Stigge, M., & Yi, W. (2011, November). Effective and efficient scheduling of certifiable mixed-criticality sporadic task systems. In Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd (pp. 13-23). IEEE.

[6] Bate, I., Burns, A., & Davis, R. I. (2017). An enhanced bailout protocol for mixed criticality embedded software. IEEE Transactions on Software Engineering, 43(4), 298-320.

[7] Baruah, S. K., Burns, A., & Davis, R. I. (2011, November). Response-time analysis for mixed criticality systems. In Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd (pp. 34-43). IEEE.

[8] Ekberg, P., & Yi, W. (2014). Bounding and shaping the demand of generalized mixed-criticality sporadic task systems. Real-time systems, 50(1), 48-86.

[9] Burns, A., & Davis, R. I. (2017). A survey of research into mixed criticality systems. ACM Computing Surveys (CSUR), 50(6), 82.

[10] Fürst, S., Mössinger, J., Bunzel, S., Weber, T., Kirschke-Biller, F., Heitkämper, P., & Lange, K. (2009, October). AUTOSAR–A Worldwide Standard is on the Road. In 14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden (Vol. 62, p. 5).

[11] Hesham Hussien, Eman Shaaban and Said Ghoniemy. "Adaptive Hierarchical Scheduling Framework for TiRTOS". In The International Journal of Embedded and Real-Time Communication Systems (IJERTCS), Volume 10, Issue 1, Article 7, 2019. 121117-045223.(in press)

[12] Bate, I., Burns, A., & Davis, R. I. (2015, July). A bailout protocol for mixed criticality systems. In Real-Time Systems (ECRTS), 2015 27th Euromicro Conference on (pp. 259-268). IEEE.

[13] Mok, A. K., Feng, X., & Chen, D. (2001). Resource partition for real-time systems. In Real-Time Technology and Applications Symposium, 2001. Proceedings. Seventh IEEE (pp. 75-84). IEEE.

[14] Joseph, M., & Pandya, P. (1986). Finding response times in a real-time system. The Computer Journal, 29(5), 390-395.

[15] Prisaznuk, P. J. (1992, May). Integrated modular avionics. In Aerospace and electronics conference, 1992. naecon 1992., proceedings of the ieee 1992 national (pp. 39-45). IEEE.

[16] Baruah, S. (2012, June). Certification-cognizant scheduling of

tasks with pessimistic frequency specification. In Industrial Embedded Systems (SIES), 2012 7th IEEE International Symposium on (pp. 31-38). IEEE.

[17] Baruah, S. (2014). Implementing mixed-criticality synchronous reactive programs upon uniprocessor platforms. Real-Time Systems, 50(3), 317-341.