A Super Scheduler Model for Hierarchical Real-Time Systems with Capability of Urgent Tasks Scheduling

Amin Enayatzare
Dependable Distributed Embedded Systems Laboratory
Ferdowsi University of Mashhad
Mashhad, Iran
a.enayatzare@gmail.com

Yasser Sedaghat
Dependable Distributed Embedded System Laboratory
Ferdowsi University of Mashhad
Mashhad, Iran
y_sedaghat@um.ac.ir

Abstract- The design of real-time systems for safety-critical applications depends heavily on the normal operation of system, in critical conditions. In these applications, among of real-time tasks, a critical task must be immediately scheduled at its arrival time immediately; otherwise, it leads to a system failure and disasters in safety-critical applications. A major problem in real-time systems included critical tasks, is unpredictable arrival of these tasks. To resolve the problem, a kind of scheduler, called "super scheduler", is employed. The problem can be more complex, in a hierarchical real-time system. A hierarchical real-time system consists of several real-time sub-systems, called "components" .Hence, using super scheduler for each component of the system, needs to special considerations. On the arrival of a critical task, the super scheduler preempts the currently running tasks and alters the priority of all existence tasks. When the critical task is completed, the preempted tasks are executed in their new priority order. This guarantees the completion of the critical and almost all other non-critical tasks before their deadlines, and therefore the stability of the component. To guarantee the stability of a hierarchical real-time system, all its components should be stable. This paper presents a model to guarantee the stability of a hierarchical real-time system included a critical task in each component. Moreover, a fault tolerance method has been applied for all components. Evaluation results show that the proposed technique improves the stability of a hierarchical real-time system included critical tasks by decreasing the number of tasks which miss their deadline.

Keywords: Super scheduler, Critical task, Real-time Scheduling, Hierarchical real-time systems, Safety-critical applications.

I. INTRODUCTION

Today, real-time systems as applicable computing paradigms affect and control our lives, ubiquitously. These systems have critical and vital requirements, especially in case of safety-critical applications, which should be considered. A safety-critical application is an application whose failure might endanger human life, lead to substantial economic loss, or cause extensive environmental damage [1]. Traditional areas that have been considered the home of safety-critical applications include avionics, automotive, medical care, and industrial automation.

In a real-time system, each task should be scheduled in order that to be completed in a specified time limit, called deadline. Among of all tasks, a critical task must be completed before its deadline; otherwise, it leads to a system failure and disasters in safety-critical applications [2], [3]. Hence, a critical task must be immediately scheduled at its arrival time. A major problem in real-time systems included critical tasks, is unpredictable arrival of these tasks. To resolve the problem, a kind of scheduler, called "super scheduler", is employed [3]. A super scheduler alters the priorities of tasks such that the highest priority is given to the critical task. Afterward, it schedules all tasks based on their new priorities. Consequently, the duty of a super scheduler is handling and change scenario of tasks execution in the case of critical tasks arrival. Due to unpredictability of arrival time of critical tasks, the super scheduler should be designed to be dynamic, flexible, and adaptable [3].

Nowadays, modern real-time applications, such as avionics, are more complex such that a simple real-time system cannot satisfy the requirements of these applications. To cope with the complexity of modern real-time systems, these systems can be independently developed and interconnected to form a System-of-Systems (SoS) [4]. A widely used model in the design of a SoS, is a hierarchical model [5]. In a hierarchical real-time system-of-systems, there are several real-time sub-systems, called components. Each component, like a simple real-time system, has several tasks and a scheduling algorithm, individually. Components in a hierarchical real-time system-of-systems are arranged in multi levels. In the first level, there is a "root component". The root component may have several child components, in the next level. Hence, a component in a middle level of hierarchy has a parent component and may have several child components. Components in the same levels of the hierarchical model employ shared computational resources, which are assigned from the parent of the components [5]. Moreover, a child component should be interacted with its parent component.

In this paper, a super scheduler technique to handle critical tasks is proposed for a hierarchical real-time system-of-systems. On the arrival of a critical task, each component of the system uses the proposed super scheduler to schedule its real-time tasks, individually. Employing the proposed technique guarantees the stability of the component, which encounters a critical task, in a hierarchical real-time system. The stability of a real-time component can be guaranteed if the number of component tasks which miss their deadlines, is minimum [3]. The stability of a hierarchical real-time system is guaranteed if the stabilities of all its components are guaranteed.

The remainder of the paper is organized as follows: Section II presents an overview of previous studies for scheduling issues in safety-critical real-time systems. In, Section III to cope with the scheduling problem of critical tasks in hierarchical real-time systems, the proposed super scheduler is described. Section IV includes the experimental results and finally conclusions are given in Section V.

II. PREVIOUS STUDIES

A. *Related Work*

Among of the task scheduling algorithms, the hybrid scheduling model, commonly have proposed for real-time and non-real-time tasks [6], [7]. Hence, a hybrid scheduling model for hard, soft and non-real-time tasks based on two-level hierarchical scheduling architecture is proposed in [8]. In the top-level of this hybrid scheduling model, a constant utilization server is created for each real-time scheduling policy. Moreover, the model has a total bandwidth server for all the non-real-time tasks which schedule by time sharing scheduling policy. The bottom-level scheduler is the operating system scheduler which responsible for scheduling all the servers. This hybrid scheduling is appropriate for the applications that consist of real-time and non-real-time parts.

An evaluation of dynamic tasks execution, under scheduling models like hybrid, static and dynamic scheduling is presented in [9]. The evaluation results show that the hybrid scheduling model can simplify selecting the most appropriate design model of the system tasks scheduling. The hybrid scheduling model guarantees that real-time tasks are completed in their deadlines. Hence, this model is appropriate for the real-time systems consist of hard real-time tasks [10], [11]. Moreover, this model can save the real-time system resources.

A scheduling method, called fast response time scheduling[1] algorithm is proposed in [12]. This hybrid algorithm is designed to fully take the advantages of Event-based scheduling (EBS) and Run-based scheduling (RBS) algorithms [13], [14]. In comparison to the static scheduling algorithms, using FRTS algorithm can improve the average response time of real-time tasks of the system, significantly.

A hybrid model containing static and dynamic methods for real-time tasks scheduling, is proposed in [2]. This hybrid scheduler comprises two phase architectural model. The first phase included an offline scheduler for real-time tasks of the system, called "fixed scheduler". Moreover, to achieve fault-tolerance for the system, a redundancy method is employed [15]. Finally, in the second phase, the hybrid scheduler is embedded in a scheduling model, called "super scheduler". To handle critical tasks, the super scheduler alters the priorities of tasks such that the highest priority is given to the critical tasks. Using this scheduling model guarantees the stability of the system, in case of safety-critical applications.

In addition to typical scheduling algorithms, the super scheduler model is employed to schedule real-time tasks in arrival of critical tasks. A suitable technique for design of the real-time tasks scheduling is proposed in [3]. Moreover, the method can evaluate the system behavior in case of safety-critical applications. The method is also appropriate for large real-time systems which consist of soft and hard real-time tasks. Thus, using this method can guarantee the stability of a large real-time system, on the arrival of critical tasks.

A hierarchical framework contained real-time components are proposed in [5]. Each component of this hierarchical framework is a real-time system which has a set of real-time tasks. These set of tasks can be scheduled with RM or EDF algorithms. Moreover, each component has a fault model for its set of tasks, individually. Under fault conditions of the components tasks, the faults can be detected and recovered [16], [17]. Thus, the task recovery methods, such as task re-execution, and forward/backward recovery can be employed in the components tasks.

According to above, the presence of critical tasks is not considered in the hierarchical real-time systems. Further, to use of task recovery methods, any of faults should be detected at the end of a real-time task execution. Hence, to handle critical tasks a fault-tolerance hierarchical real-time system that contains the super scheduler can be defined.

B. *Problem Definition*

Assume that each component of a hierarchical real-time system has a set of real-time tasks. The problem is studied under overloads conditions of real-time tasks that scheduled by a typical scheduler. The arrival of unpredictable critical tasks is the most common cause of these overloads conditions. These critical tasks will lead to the scheduling scenario changes. Hence, most of the real-time tasks should be complete in their deadlines. Moreover, the stability of the hierarchical real-time system should be guaranteed.

III. PROPOSED TECHNIQUE

A. *System Model*

A hierarchical real-time system[2] is a real-time system-of-systems (SoS). This composed of multiple levels of real-time sub systems, called components with a hierarchy. Each component, like a simple real-time system, has several tasks and a scheduling algorithm, individually. Moreover, each component consists of multiple concurrent processors to execute its real-time tasks. The components in the same levels of hierarchy

[1] FRTS Algorithm

[2] HRTS

employ shared computational resources, which are assigned from their parent components. Hence, the basic unit of the HRTS is a component which is defined by (W, R, A) as follows:

- W: workload or a set of tasks in a component
- R: resource model supported from a upper level
- A: scheduling algorithm of W for a given resource model R [5].

A workload set W composed of periodic real-time tasks which execute on the concurrent processors of the component. A resource model R specifies the amount of resource allocated from the parent component. In order that, the periodic resource model $\Gamma(\Pi, \Theta)$ is employed [18], [19], where Π is the resource supply period and Θ is the resource supply time per period. Hence, a component with the resource model $\Gamma(\Pi, \Theta)$ is guaranteed to be supplied with Θ resource time every Π time units from its parent component.

Moreover, the HRTS has an interface model to interaction between child and parent components. Hence, each parent component takes the real-time resource requirements of its child components. Afterward, requested resources of the child components are supplied by their parent components. The interface model I is defined by (P,E) [5], where P is period and E is resource supply time. For a child component, if the resource with the amount of E is supported per the period P from its parent component, then the feasibility condition is guaranteed.

B. Fault Handling Model

In the proposed model assume that each component of the system has a critical manager section. On the arrival of a critical task or any fault detection, the critical manager should handle the critical conditions. On the arrival of critical tasks, the super scheduler model which consists of three phases is employed. In first phase, the scheduling algorithm of each component, schedules the component tasks with an offline or fixed scheduler, like RM[3].

In the second phase, to achieve fault tolerance for the system components, a hardware redundancy method [20] is employed. A primary-backup redundancy is used for the concurrent processors of the system components. The backup copies can be overlapped to reduce the number of backup processors. Moreover, these backup copies of concurrent processors are preserved in the critical manager section of each component. This phase is, called "dynamic scheduler".

In the third phase, the super scheduler is used to schedule the real-time tasks of the components such that, on the arrival of a critical task, the component automatically performs the context switching of fixed scheduler to the super scheduler. This phase is called "critical scheduler".

C. Task Model

The timing properties of a given task of the components are as following [21]:

- *Release time (ready time)(r_i)*: Time at which the task is ready for processing.
- *Deadline (d_i)*: Time by which execution of the task should be completed.
- *Execution time (e_i)*: Time taken without interruption to complete the task, after the task is started.
- *Priority (ζ_i)*: Relative urgency of the task.
- *Period (P_i)*: A periodic task τ_i is a task recurring at intervals of time. Period P_i is the time interval between any two consecutive occurrences of task τ_i.

With respect to timing properties of the tasks, the fairness of the schedule defined [22] as follow: The sum of release time and execution time of a task should be less than the deadline of the task. To have a precise schedule, each task T_i from task set T should have its

$$r_i+e_i \leq d_i \qquad (1)$$

D. Scheduler Model

The proposed scheduling model is composed of three phases: fixed scheduler, dynamic scheduler and the critical scheduler. Figure 1, shows a schematic view of the proposed model.

Assume that each component contains 'n' primary and 'm' backup concurrent processors. One of processors of the component uniquely considered for critical task execution which, called "critical processor". Until arriving of critical task, the critical processor of each component will be idled. The processors of each component have different priorities, such that the priority of critical processor is higher than other processors. Moreover, the priorities of other processors are set based on the system requirements.

With respect to Figure 1, the proposed scheduling model has a segmented shared memory for the set of tasks of each component. It can be used to take the advantage of simultaneous sharing of memory by multiple tasks [24].

The aggregation of the first and second phases of scheduling model is called "hybrid scheduler". The hybrid scheduler embedded in the super scheduler model which consists of three mentioned phases. Consider a component of system consisting set of tasks, T= $\{T_1, T_2, T_3,\ldots,T_n\}$

[3] Rate Monotonic

and a critical task CT with unpredictable arrival time. Until the arrival of critical task CT, task T_1 to T_n can be scheduled by the hybrid scheduler. On the arrival of the CT, the component scheduler switches to super scheduler, immediately. Then, the super scheduler preempts the currently running tasks and alters the priority of all existence tasks. When the critical task CT is completed, the preempted tasks are executed in their new priority order. Here, a few low priority tasks cannot meet its deadline and will be discarded, since they don't cause any system failure.

The stability ratio of a real-time system using super scheduler, should be greater than 0.7 [2], [3]. Hence, the super scheduler guarantees the stability of system even in the case of critical tasks arrival. Therefore, the stability ratio of each component of HRTS can be as follow:

$$R_{Stb} = \frac{N-n_M}{N} > 0.7 \quad (2)$$

The deadline miss ratio of components tasks as an evaluation criterion can be as follow:

$$R_{DM} = \frac{n_M}{N} \qquad (3)$$

n_M: number of low priority tasks which missed its deadline.

N: Total number of tasks super scheduled.

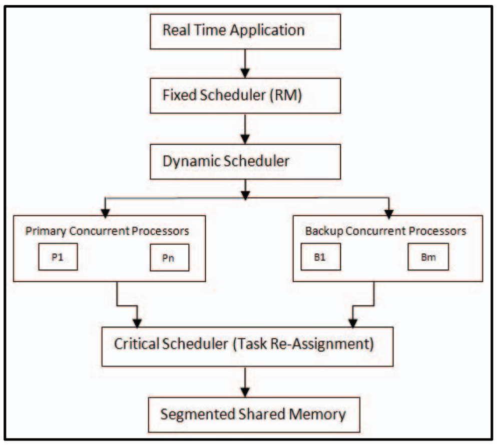

Real Time Application
Fixed Scheduler (RM)
Dynamic Scheduler
Primary Concurrent Processors
P1
Pn
Backup Concurrent Processors
B1
Bm
Critical Scheduler (Task Re-Assignment)
Segmented Shared Memory

Fig.1. Proposed super scheduler model

Equation (2) can be used for single/multi processor real-time systems. To guarantee the stability of each component, the stability ratio of the component should be greater than 0.7, even after the arrival of a critical task.

To guarantee the stability of a hierarchical real-time system, all its components should be stable. Hence, the child components of the system should compute their stability ratio before and after the arrival of a critical task. Then transmit these stability ratios to their parent component, using their interface model. Afterward, each parent component specifies the stability of its child component. Then, the parent components transmit their stability ratio to their upper-level component. Therefore, the stability of system components is guaranteed in each parent component, locally. Finally, the root component specifies the stability of whole system, globally. Figure 2, shows the architecture of a HRTS, included the proposed super scheduler.

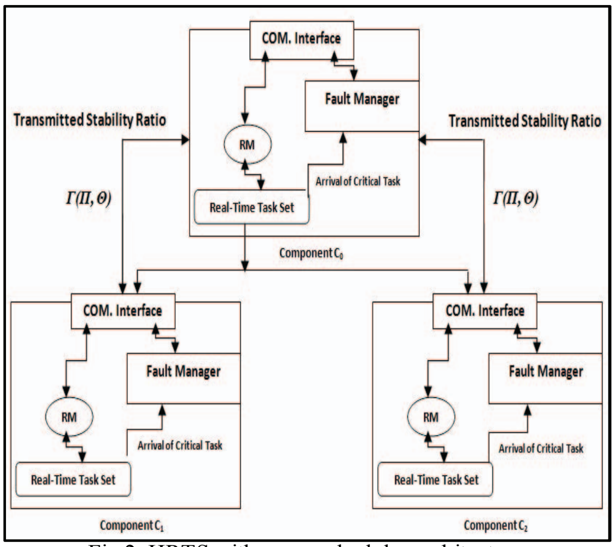
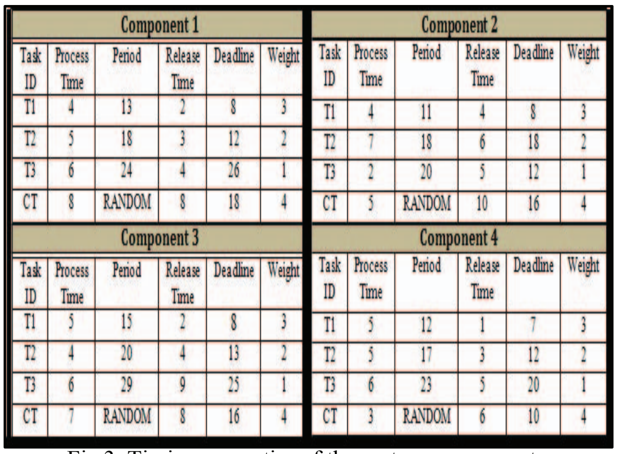


Fig.2. HRTS with super scheduler architecture

IV. EXPERIMENTAL RESULTS

Using proposed super scheduler in the HRTS was implemented by MATLAB. Furthermore, the TORSCHE toolbox was employed to define and implement of periodic real-time tasks of the components and scheduling algorithms. As for the experiment setup, it was assumed that the hierarchical real-time system consists of 4 components. Each component was consisting of 3 typical periodical real-time tasks and 1 unpredictable critical real-time task. It was considered that a real-time task composed of follow properties: Execution time, release time, period, priority and deadline such that, the task deadline $d=t_i$ means that, the task should be completed in t_i unit, after its arrival. Moreover, the priority of a critical task was higher than other typical real-time tasks. Since the arrival of the critical task was unpredictable, a random number generator function was used to generate the random arrival times of each critical tasks. Figure 3 shows the tasks properties of the considered HRTS components.

For instance, with respect to the timing properties of the Component 1, the execution sequence of real-time tasks is as <T1, T2, T3>. While in presence of critical task CT the tasks scheduling scenario will be changed. Task T1 arrives at 2 and its process time takes 4 units. On other hand, T2 should be executed after T1, but, its process time takes 5 units and the arrival of critical task CT is at 8. Thus, after two unit of execution, T2 is preempted by super scheduler. Then the priority of T2 and T3 are altered and CT starts its execution. When the CT is completed, T2 is executed in its new priority order. Task T3 executes as final task, also in its new priority order. The execution sequence in presence of critical task CT is as <T1, $T2^*$, CT, T2, T3>. Where the $T2^*$, means that on the arrival of CT, task T2 is preempted. Figure 4 shows the Gantt chart of task scheduling, using proposed super scheduler for Component 1. Hence, among of 4 tasks (T1, T2, CT, T3), deadline of one task (T2) is missed. But, as shown as in Equation (2), the stability of Component 1 is guaranteed with the stability ratio of 0.75.



Component 1					
Task ID	Process Time	Period	Release Time	Deadline	Weight
T1	4	13	2	8	3
T2	5	18	3	12	2
T3	6	24	4	26	1
CT	8	RANDOM	8	18	4

Component 2					
Task ID	Process Time	Period	Release Time	Deadline	Weight
T1	4	11	4	8	3
T2	7	18	6	18	2
T3	2	20	5	12	1
CT	5	RANDOM	10	16	4

Component 3					
Task ID	Process Time	Period	Release Time	Deadline	Weight
T1	5	15	2	8	3
T2	4	20	4	13	2
T3	6	29	9	25	1
CT	7	RANDOM	8	16	4

Component 4					
Task ID	Process Time	Period	Release Time	Deadline	Weight
T1	5	12	1	7	3
T2	5	17	3	12	2
T3	6	23	5	20	1
CT	3	RANDOM	6	10	4

Fig.3. Timing properties of the system components

By increasing the number of real-time tasks of the system and only using hybrid scheduler, the number of tasks which miss their deadlines will be increased, consequently. But as shown in equation (3), the proposed super scheduler can improve the task deadline miss ratio of the system. As for another experiment setup, It is assumed that there is a HRTS with 7 components called AC1, AC2,…, AC7, such that these 7 components consist of 5, 11, 18, 26, 32, 40 and 48 tasks, respectively. Furthermore, each of these components consists of an unpredictable critical task. It is assumed that by using proposed super scheduler, on the arrival of critical task, 1, 2, 4, 6, 9, 12 and 15 tasks missed their deadlines in AC1,…, AC7, respectively. The deadline miss ratio of these components tasks using hybrid and super scheduler concludes the follow results. In a HRTS, using proposed super scheduler is more efficient than the hybrid scheduler model. Figure 5 as an experimental result of the system, approves this result by comparison of deadline miss ratio for the assumed HRTS components tasks. The typical super scheduler model [2], [3] is proposed for centralized real-time systems but, it is not applicable for handling the critical tasks in the distributed real-time systems [25], [26]. To employ this model in the distributed real-time systems, the model should be scalable [27]. The HRTS included the proposed super scheduler model can be expanded to a real-time system which comprises large numbers of components. Furthermore, for a large HRTS, each component can be located in different geographical area. These components can interact with their parent components, using their interface models. The parent components act as local coordinator nodes [4]. Finally, the root component as a global coordinator, guarantees the stability of HRTS. Thus, the proposed super scheduler can improves the operation of typical super scheduler model such that, proposed model can be employed in distributed real-time applications.

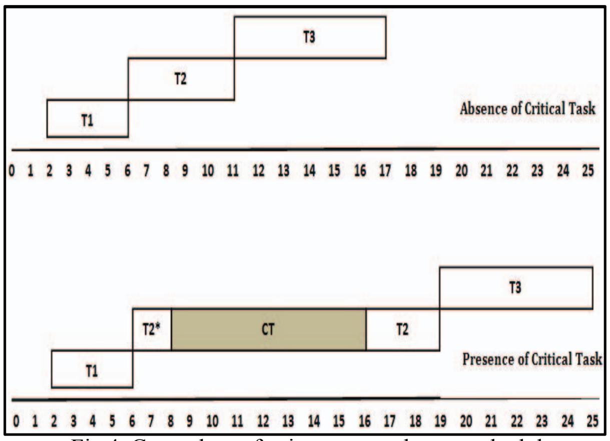


Fig.4. Gantt chart of using proposed super scheduler for component 1 tasks

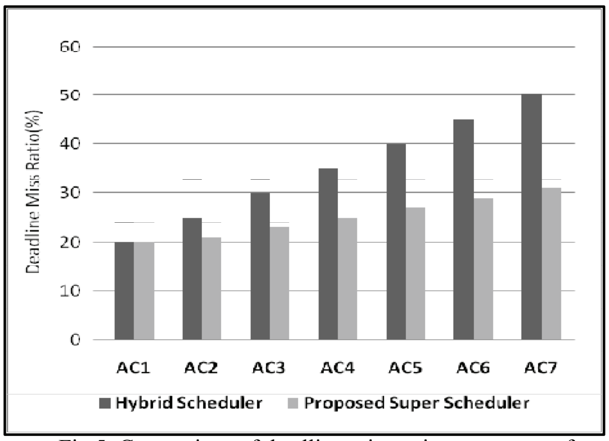


Fig.5. Comparison of deadline miss ratio percentage of assumed components tasks, using proposed super scheduler and hybrid scheduler

V. CONCLUSIONS

The fault-tolerant schedulers in case of critical tasks arrival can be employed in the hierarchical real-time systems. Among of this schedulers, using super scheduler can guarantee the stability of the real-time systems, even after critical tasks arrival. To guarantee the stability of a hierarchical real-time system, all its components should be stable. Hence, using super scheduler for each component of a hierarchical real-time system guarantees the stability of the system. Evaluation results show that employing the proposed super scheduler in each component of a hierarchical real-time system, improves the deadline miss ratio of component tasks, significantly. Consequently, employing the proposed super scheduler can improve the stability of the hierarchical real-time systems more than both of the typical and the hybrid schedulers.

VI. REFERENCES

[1] J.C. Knight, “Safety critical systems: challenges and directions”, in Proceedings of 2002 24rd IEEE International Conference on Software Engineering, pp 547 – 550, May 2002.

[2] T.R.Gopalakrishnan Nair, A. Christy Persya, “Critical Task Re-assignment under Hybrid Scheduling Approach in Multiprocessor Real-time Systems", Parallel and Distributed Computing and Systems, USA. 23rd IASTED International Conference on, vol., no., pp.130-137, December 2011.

[3] A. Christy Persya, T.R.Gopalakrishnan Nair, “Model based design of super schedulers managing catastrophic scenario in hard real-time systems”, Proceeding of 2013 International Conference on Information Communication and Embedded Systems (ICICES),pages 1149 – 1155, February 2013.

[4] H. Kopetz, Real Time Systems: Design Principles for Distributed Embedded Applications, 2nd Edition, Springer, 2011.

[5] J.Hyun, K.J. Kim, “Fault-Tolerant Scheduling in Hierarchical Real-Time Scheduling Framework”, in proceedings of 2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pages 431 – 436, August 2012.

[6] Z. Deng, J. W.-S. Liu, and J. Sun, “A Scheme for Scheduling Hard Real-Time Applications in Open System Environment”, Proceeding of 9th Euro micro Workshop on Real-Time Systems, pp.191-199, June 1997.

[7] Z. Deng, J. W.-S. Liu, “Scheduling Real-Time Applications in Open System Environment”, Proceeding of IEEE Real-Time Systems Symposium, San Francisco,vol., no., pp 308-319, December 1997.

[8] P. Tan, H. Jin, M. Zhang. A, “Hybrid Scheduling Scheme for Hard, Soft and Non-Real-time Tasks”, in Proceeding of the ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, pp 20-26, April 2006.

[9] M. Nolin, K. Hänninen, J. Mäki-turja, “Efficient Development of Real-Time Systems Using Hybrid Scheduling in Embedded Systems and Applications”, in Proceedings of the International Conference on Embedded Systems and Applications, vol.6, pp 53-59, June 2005.

[10] K. Sandström, C. Eriksson, and G. Fohler, “Handling Interrupts with Static Scheduling in an Automotive Vehicle Control System”, Proceeding of the 5th IEEE International conference on Real-Time Computing Systems and Applications, pp.158-165 , October 1998.

[11] J. M.Turja, M. Nolin, “Tighter Response-Times for Tasks with Offsets”, Proceeding of the 10th International conference on Real-Time Computing Systems and Applications, pp.127-36, August 2004.

[12] I. Yan, J. Zhang, M. Shan, "Scheduling for fast response multi-pattern matching over streaming events", Data Engineering (ICDE), 2010 IEEE 26th International Conference on, pp.89-100, March 2010.

[13] M. Liu, M. Li, D. Golovnya, E. Rundensteiner, K. Claypool, “Sequence pattern query processing over out-of-order event streams”, proceeding of the25th IEEE International Conference on Data Engineering, pp.784-795, March 2009.

[14] D. Carney, U. Cetintemel, A. Rasin, S. Zdonik, M. Cherniack, and M. Stonebraker, “Operator scheduling in a data stream manager”, Proceedings of the 29th international conference on Very large data bases-Volume 29, pp. 838-849, October 2003.

[15] A.C. Persya, T.R.G.Nair, ”Fault Tolerance in Real-Time Multiprocessors Embedded Systems”, Fourth Innovative Conference on Embedded Systems, Mobile Communication and Computing, pp.234-238 , July 2009.

[16] G.M. Tchamgoue, K.H. Kim, Y.K. Jun, W.Y. Lee, “Hierarchical real-time scheduling framework for imprecise computations”, Proceeding of IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing, pp. 273–280, December 2010.

[17] C.M. Krishna, A.D. Singh, “Reliability of check pointed real-time systems using time redundancy”, IEEE Trans On Reliability, vol.42, pp 427–435, September 1993.

[18] I. Shin, I. Lee, “Compositional real-time scheduling framework”, proceeding of the 25th IEEE Real-Time Systems Symposium, pp. 57-67, December 2004.

[19] I. Shin, I. Lee, “Compositional real-time scheduling framework with periodic model”, ACM Transactions on Embedded Computing systems, vol.7, no.30, pp.2-13, April 2008.

[20] A. Ejlali, B.M. Al-Hashimi, P. Eles, “Low Energy Standby-sparing for Hard Real-Time Systems”, IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 3, pp. 329-342, March 2012.

[21] M. Joseph, Real-time Systems: Specification, Verification and Analysis, Prentice Hall, 1996.

[22] J.W.S. Liu, Real-Time Systems, 2nd Edition, Pearson Education, 2005.

[23] C.M. Krishna, K.G. Shin, Real-Time Systems, McGraw-Hill, 1997.

[24] M. Naghibzadeh, P. Neamatollahi, “Efficient Semi-Partitioning and Rate-Monotonic Scheduling Hard Real-Time Tasks on Multi-Core Systems”, IEEE International Transaction Symposium on Industrial Embedded Systems, pp. 85-88, June 2013.

[25] M. Dinatale, J.A. Stankovic, “Dynamic End-to-End Guarantees in Distributed Real-Time Systems”, Proceeding of IEEE Real-Time Systems Symposium, San Juan , pp. 216-227, Dec 1994.

[26] J.A. Stankovic, H.Tian, T. Abdelzaher, M.Marley, “Feedback Control Scheduling in Distributed Real-Time Systems”, Proceeding of 22nd IEEE Real-Time Systems Symposium, pp. 59-70, Dec 2001.

[27] P. Jogalekar, M. Woodside, “Evaluating the Scalability of distributed systems”, IEEE Transaction on Parallel and Distributed systems, vol. 11, no. 6, pp. 589-603, Jan 2000.

Filename:	A Super Scheduler Model for Hierarchical Real-Time Systems with Capability of Critical Tasks Scheduling_ Amin Enayatzare
Directory:	C:\Users\Amin\Desktop
Template:	C:\Users\Amin\AppData\Roaming\Microsoft\Templates\Normal.dotm
Title:	
Subject:	
Author:	CE
Keywords:	
Comments:	
Creation Date:	7/22/2014 1:17:00 AM
Change Number:	41
Last Saved On:	9/23/2014 12:33:00 PM
Last Saved By:	CE
Total Editing Time:	320 Minutes
Last Printed On:	9/23/2014 6:31:00 PM
As of Last Complete Printing	
Number of Pages:	6
Number of Words:	4,254 (approx.)
Number of Characters:	24,254 (approx.)