A Schedulability Analysis for Weakly Hard Real-Time Tasks in Partitioning Scheduling on Multiprocessor Systems

Habibah Ismail, Dayang N. A. Jawawi
Software Engineering Department,
Faculty of Computing,
Universiti Teknologi Malaysia,
Johor, Malaysia
habibahisma@gmail.com, dayang@utm.my

***Abstract*—The importance of missing a deadline is one of the classifications of a real-time tasks or systems. A weakly hard real-time system is a new generation for a real-time system in which some degree of missed or losses deadlines are tolerated occasionally but the missing of deadlines has to be stated precisely. For a traditional real-time system, timing requirements in hard real-time are very restrictive, must meet all the tasks deadlines. Meanwhile, in soft real-time timing requirements, the requirements are too relaxed because there are no guarantees can be given for the deadline, whether it is met or missed. As a systems demand complex and significantly increased functionality, multiprocessor scheduling has been given attention and taken into consideration. In fact, within the multiprocessor, the predictability problems seem even harder. Thus, in order to deal with the problem, the partitioned multiprocessor scheduling techniques with solutions for the bin-packing problem, named R-BOUND-MP-NFRNS (R-BOUND-MP with next-fit-ring noscaling) combining with the exact analysis, called hyperperiod analysis and deadline models; weakly hard constraints and μ-pattern are present to provide weakly hard real-time guarantees under static priority scheduling algorithm. We prove that the proposed approach provides predictable weakly hard real-time tasks through validation and simulation results.**

Keywords—weakly hard real-time systems; schedulability analysis; multiprocessor systems; partitioning scheduling; hyperperiod analysis

I. INTRODUCTION

Every task in the system is completed in period of time is referred to a real-time system. A real-time system, should be predictable and ensure that all task time period requirements assumptions always be met. Guaranteeing temporal correctness is the main goal of real-time systems analysis where the verification of the task priority which includes the no task deadline is missed and allowing the miss deadlines for the tasks. The temporal correctness is depending on the task that has being scheduled. We need to utilize real-time scheduling algorithms and derive schedulability tests to validate temporal correctness.

The timing constrained requirements are the direct input for the scheduling analysis algorithms. One of the major benefits of schedulability analysis is the ability to detect any deadlines in given task using a predicting the timing behaviour of a set of real-time systems [1]. It is used in many different ways such as at design time or at run-time. In order to do the schedulability analysis and design for real-time systems, related scheduling parameters, such as task execution time and task period are important to be obtained.

The timing constraints are being specified in terms of the deadlines activity. The classification of real-time systems or tasks divided into two categories, based on the "seriousness" of deadline misses – hard real-time tasks and soft real-time tasks [2]. The consequences of a deadline miss of a hard real-time task can be prohibitively expensive whereas "some" deadline misses are tolerate for soft real-time task systems. An example of a hard real-time system is an automobile braking system. When the driver pressed the brakes, the automobile can meet with an accident, if the systems are not appropriately respond.

On the other hand, consider an online transaction system as an example of a soft real-time system in which some laxity of task deadlines are tolerated; the user does not mind if while the processing of their transaction, a little delay happens as long as they are within "acceptable" limits. Usually, hard tasks co-exist with soft tasks; thus, it means that most hard real-time tasks are not that hard actually. It may be the case that the real-time application consists of a mix of hard and soft real-time tasks, and it called weakly hard real-time system. Also known as hard systems that is not strongly hard.

A typical example of systems with weakly hard real-time requirements is multimedia systems because some delay during execution is acceptable in that system and it is unnecessary to meet all the tasks deadlines as long as the misses (or deadlines) are specified precisely. In a weakly hard real-time system, the number of deadlines that may be missed can be specified. This makes a weakly hard real-time system stronger than a soft real-time system.

978-1-4799-5439-1/14/$31.00 ©2014 IEEE

In weakly hard real-time scheduling, usually the scheduling analysis was focused in the uniprocessor. As a systems demand complex and significantly increased functionality, the multiprocessor scheduling identify to be one of the optional method instead of uniprocessor. In this paper, weakly hard real-time system has been studied rather than hard and soft real-time systems because "weakly hard" precisely define and specify the distribution of missed and met deadlines for the tasks during a given time period and also, pattern of deadlines while knowing when and how many deadlines may be missed.

Research on multiprocessor scheduling has been mainly focused on guaranteeing strict deadline observance. The classification of multiprocessor scheduling is divided into partitioned and global (or some combination thereof) [3]. In partitioning scheduling approach, every task assigned in the processor is permitted to be executed with no migration with each processor schedule the assigned tasks independently with a uniprocessor scheduling algorithm. And, in the partitioning multiprocessor scheduling, usually, divided a task set into m (m is referred to the number of processors) groups. Then, assigned tasks in each group to one of the processor, and all the processors used are independently scheduled. In global scheduling, tasks may migrate among processors or in other words, on different processor a task is permitted to be executed and scheduled from a single priority queue. Also, in global multiprocessor scheduling, need to allocate the available processors to the highest priority task in the ready queue.

The aim of this paper is to contribute towards improved real-time scheduling by providing a schedulability analysis of periodic and pre-emptive tasks that are constrained by weakly hard constraints on multiprocessor at design time. Static priority scheduling is considered in order to handle overloads. In static schemes, processes are assign to a fixed priority. Fixed priority scheduling is relatively easy to be implement and requires less overhead than dynamic schemes. The objective of the paper is to schedule a weakly hard real-time tasks on multiprocessor and to guarantee the system is predictable.

The paper is then structured as follows: Section II presents the related works. Then, the partitioning scheduling and hyperperiod analysis is placed in Section III. After that, autonomous mobile robot system case study is in Section IV. Section V discusses a schedulability analysis in partitioned approach. In Section VI, this paper draws some conclusions and drafts remaining ongoing work.

II. RELATED WORKS

There have been some efforts to schedule weakly hard real-time tasks on multiprocessor systems. Wu and Jin proposed the classical weakly hard real-time scheduling algorithms, namely Distance Based Priority (DBP) to apply into multiprocessor applications to guarantee QoS of both hard real-time tasks and multimedia streams even under overload conditions [4]. In fact, the DBP algorithm originally was introduced by [5] on uniprocessor system.

Another work is done by Kong and Cho wherein they design a new dynamic scheduling algorithm known as the Guaranteed Multiprocessor Real-Time Scheduling (GMRTS-MK) algorithm for (m,k)-firm constrained tasks on homogenous multiprocessors [6].

Several static and dynamic approaches for weakly hard real-time tasks on uniprocessor system have been proposed. Quan and Hu presented an approach based on an offline scheduler of periodic tasks called a heuristic technique to improve the (m,k)-patterns [7]. In the approach, each task is divided into two sets; mandatory and optional. The mandatory tasks are scheduled according to their pre-defined priorities, while optional tasks are assigned to the lowest priority. They also present a fitness function from genetic algorithm (GA) as an estimated metric to minimize the execution interference between tasks. They propose a sufficient condition, a polynomial time algorithm to test the schedulability of real-time systems. However, a more precise fitness function need to be constructing for a task set with weakly hard constraints and in order to predict the schedulability of such a task set, tighter sufficient conditions must be searched.

A new fixed-priority scheduling algorithm basing on Rate Monotonic Algorithm (RMA) scheduling policy is presented by Ming and Hai which divides the task into pre-emptive and optional to reflect its urgency basing on weakly hard real-time parameter, and the algorithm can modify the priority of task dynamically during scheduling [8]. It makes the algorithm more flexible. However, the approach is easy to implement and they introduces very low scheduler overhead.

Ould-Sass, Chetto and Queudet propose to modelize the execution requirements of periodic tasks with the Skip-Over approach and the Deadline Mechanism, both integrated in a novel QoS model called Black-Grey-White model, BGW for short [9]. They deal with the problem of scheduling sets of firm periodic tasks capable of handling exceptions such as faults and overload.

Based on the review, we present static offline scheduling since it can ensure the predictability of system behaviour. Hence, offline processor assignment for multiprocessor, called the partitioned scheduling algorithm is chosen for our schedulability analysis because the advantage of that approach is at any runtime overhead caused by migrations of task can be avoided. Also, the advantages of this approach are its efficiency and simplicity. In most cases, if the task set is fixed and known a priori, the partitioning approach is becoming the most appropriate solution.

III. THE PARTITIONING SCHEDULING AND HYPERPERIOD ANALYSIS

In the partitioning approach [3], no migration is allowed, where tasks are statically partitioned and allocated to the processors. Need to choose a processor for all tasks and then run local scheduler on each processor with no migration. Also, it is known as offline processor assignment. At design time before execution, priority is to statically assign a set of periodic tasks to a set of processors. The main advantage of partitioning approaches is that it reduces a multiprocessor

scheduling problem to a set of uniprocessor ones and partitioning approaches are widely used by system designers.

Each task having its own dedicated processor, due to the partition method divides tasks into partitions. Therefore many heuristic have been proposed for partitioning, the bin packing algorithm is a majority chosen versions to be used [10]. In bin-packing algorithms, the test on the schedulability is required to determine whether the task allocation in the processor is needed. As a schedulability test, the following equation [10] is to use the knowledge that:

$$\sum_{i=1}^{Ci} \frac{}{Ti} \leq \eta_\rho \,.\, (2^{1/}\eta_\rho - 1) \qquad (1)$$

Note that the rate monotonic algorithm is used to schedule tasks on processor p. Thus, p is defined by the total of task assigned to processor p which denote by n_p. In [11], algorithm R-BOUND-MP-NFRNS (R-BOUND-MP with next-fit-ring noscaling) was used to partition the scheduling algorithm which derives from the dual-processor scheduling algorithm, R-BOUND-MP (combined R-BOUND with a first-fit-bin packing algorithm), that exploits R-BOUND [12]. R-BOUND is a uniprocessor scheduling algorithm for partitioned scheduling. The R-BOUND-MP-NFRNS requirements are elaborate as follows:

- The sorting of tasks in the ascending order of periods where the first task needed to be considered is the task with the shortest or smallest period.
- Each uniprocessor is use Equation 1 in a schedulability test.
- The next-fit-bin-packing algorithm was used to assign each task.
- The task was assigned to processor 1 due to the failure of the task assign into processor p.

One way to solve predictability problem in schedulability analysis of weakly hard real-time tasks is to perform the exact analysis such as hyperperiod analysis combined with partitioned scheduling algorithm. A hyperperiod analysis is used to calculate the least common multiple for the tasks. It is possible to use hyperperiod analysis on multiprocessor scheduling because of using periodic task with deadline and preemptive fixed priority scheduling.

The higher order period or the hyperperiod, h_i consist the number of invocations of a task in the hyperperiod at level i, $a_i = h_i/T_i$ [13]. The least common multiple tool is used to get the values of the hyperperiod. The equation of the hyperperiod h_i is given by [13]:

$$h_i = lcm\{T_j \mid \in hep(\tau_i)\} \qquad (2)$$

IV. AN AUTONOMOUS MOBILE ROBOT (AMR) SYSTEM CASE STUDY

We chose the AMR case study by [14] because it consists of hard and soft tasks. Thus, it is unnecessary for the system to meet the entire task deadlines as long as the misses (or deadlines) are spaced distantly/evenly or, in other words, is weakly hard real-time tasks. We first present a task set derived from the AMR system, that consist of seven tasks and all the tasks are listed in Table I.

TABLE I. THE TASK SET OF AMR CASE STUDY

Task	T_i	D_i	C_i
MobileRobot	50	50	1
motorctrl_left	50	50	20
motorctrl_right	50	50	20
Subsumption	80	80	1
Avoid	90	90	17
Cruise	90	90	1
manrobotintf	95	95	16

The AMR used in the case study is capable of traversing in a structured environment and has a differential drive wheeled mobile robot. The goal of the robot software is to control the movement of the robot while avoiding obstacles in its environment.

In the parameters, T_i is the period of the task, and D_i refers to the relative deadline of the task that must be finished. Every task is periodic and we assumed that $D_i = T_i$. C_i represents the worst case execution time of the tasks.

V. A SCHEDULABILITY ANALYSIS IN PARTITIONED APPROACH

A schedulability tests or analysis is a condition in order to know whether a task set meets its deadlines or not. Usually, schedulability tests for partitioned multiprocessor task systems (combining well-known uniprocessor scheduling techniques with solutions for the bin-packing problem). The system behaves in partitioned scheduling as explained in details as follows.

The processor is assigned with each task in order to assign a local (for a processor) with static priority. *MobileRobot* has the highest priority because the task priority was decreasing in each processor. Consider seven tasks in Table I, where these tasks needed to be scheduled using 2 processors with R-BOUND-MP-NFRNS. Furthermore, the algorithm is responsible to sort the task periods in ascending order. On this analysis, processor 1 is the current one with the tasks has been assigned in order. Moreover, *MobileRobot* has been selected and assigned into processor 1. Afterwards, *motorctrl_left* has been tried to assign into processor 1, which is successful due

to the utilization sum for these two tasks is 0.42. An attempt is made to assign the *motorctrl_right* into processor 1 afterwards and succeeds due to the utilization sum of these three tasks which is 0.82. Later, the *Subsumption* is selected to assign into processor 1, and it succeeds as the sum of utilization of this tasks is 0.8325.

The task named *Avoid* is attempted to be assigned into Processor 1 where it fails due to the utilization sum of this tasks is 1.0225 which resulting the task is assigned into processor 2. Next, processor 2 is identified as the current processor. The task named *Cruise* is attempted into processor 2 and succeeded as the utilization sum of this task is 1.0325. Afterwards, the processor 2 has been assigned with task named *manrobotintf* which the task assign is fail due to the utilization sum of three tasks is 1.2025. Hence, it is assigned to processor 1. Table II shows which processor of each task is assigned.

Later, Fig. 1 shows a timing diagram which has been designed for a partitioned static priority scheduling algorithm along with the utilization bound U, wherein $U_i = C_i/T_i$. From the figure, the reader should realize that task *manrobotintf* missed its deadline by five time units. This is made task *manrobotintf* unschedulable. Hence, to ensure the behaviour of the whole system is predictable, hyperperiod analysis and weakly hard temporal constraints can be used to guarantee that both deadlines and constraints are satisfied if there have one or more tasks missed its deadline. In order to guarantee that task *manrobotintf* meet its deadlines and satisfy its weakly hard temporal constraints, an exact analysis is perform to make the tasks predictable. In order to show that exact number of deadlines that can be missed, hyperperiod analysis and weakly hard temporal constraints are used. Table III presents the hyperperiod analysis for each task.

TABLE II. TASK PARAMETERS OF THE TASK SET AND PROCESSORS

Task	T_i	D_i	C_i	U_i	P_i
MobileRobot	50	50	1	0.02	1
motorctrl_left	50	50	20	0.4	1
motorctrl_right	50	50	20	0.4	1
Subsumption	80	80	1	0.0125	1
Avoid	90	90	17	0.19	2
Cruise	90	90	1	0.01	2
manrobotintf	95	95	16	0.17	1

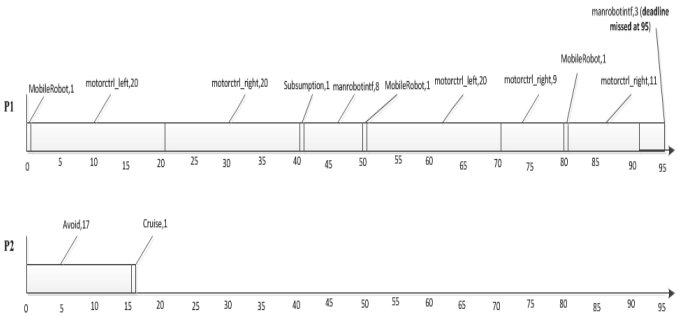

Fig. 1. Timing diagram of partitioned scheduling

From the results in Table III, even though task *manrobotintf* missed its deadline at the critical instant, but by using hyperperiod analysis, we can specify the number of deadline missed for that task.

The Fig. 2 shows the utilization bound at each invocation within the hyperperiod at priority level 7. The task is invoked $\alpha_7 = 10$ times within the hyperperiod at level 7.

The following Table IV shows the exact distribution of the missed and met deadlines for invocations 1 to 10 of task *manrobotintf*.

As can been seen task *manrobotintf* missed its deadline at its fourth and ninth invocations. Using weakly hard constraints, we can precisely specify the number of deadline met and missed for the task.

So, task *manrobotintf*'s μ-pattern would be 11101 11101. A 0 represents a deadline missed and a 1 a deadline met. It can miss it at most 2 times during its hyperperiod, (2 times every 10 invocations).

TABLE III. THE HYPERPERIOD ANALYSIS

Task	T_i	D_i	C_i	h_i	a_i
MobileRobot	50	50	1	50	1
motorctrl_left	50	50	20	50	1
motorctrl_right	50	50	20	50	1
Subsumption	80	80	1	400	5
Avoid	90	90	17	450	5
Cruise	90	90	1	450	5
manrobotintf	95	95	16	950	10

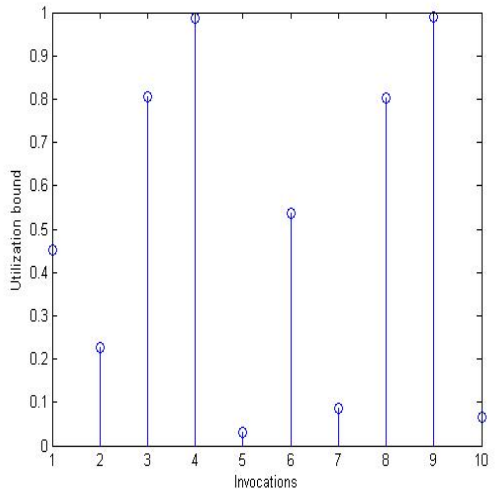

Fig. 2. Invocation of *manrobotintf* in the hyperperiod

TABLE IV. THE EXACT DISTRIBUTION OF THE TASK

Task *Manrobotintf*	
Invocations	***μ-pattern***
1 - 10:	11101 11101

Thus, the weakly hard constraint for task *manrobotintf* is defined as (4,5) constraint. Checking *manrobotintf*'s weakly hard constraint shows that it is satisfied, despite the miss and we can concluded that the system is weakly hard schedulable. The main objective of weakly hard constraints is to guarantee the tasks meet their timing constraints even though during execution time there are some deadlines may be missed. Meanwhile, μ-patterns are used to determine how the deadline can be missed in terms of the consecutiveness or non-consecutiveness of such missed and met deadlines.

VI. CONCLUSIONS

As conclusion, in this paper, we presented an offline static schedulability analysis in order to schedule periodic weakly hard real-time tasks. We focused on multiprocessor systems and partitioned scheduling approach that has been used. We use R-BOUND-MP-NFRNS partitioned scheduling algorithm under the rate of monotonic algorithm in order to assign each of the tasks to the processors. Then, from the schedulability tests, task *manrobotintf* missed its deadline, thus to guarantee the system is predictable, we analysed the task using hyperperiod analysis and weakly hard temporal constraints in order to precisely knows the number of deadline met and missed for the task. From the results, it shows that although the task *manrobotintf* missed its deadline, it is weakly hard schedulable because we can guarantee both deadlines and constraints are satisfied. Therefore, it makes the system predictable and schedulable. Our experimental simulation validates that the proposed approach achieves our scheduling objective and provide a solution to the problem stated in this paper. Regarding this approach, designers can provide more predictable weakly hard real-time systems based on using multiprocessor scheduling technique with exact analysis such as hyperperiod analysis combining with deadline models or temporal constraints. Also, our approach can be useful on any multiprocessor system whose worst-case bound delays can be obtained. For future work, we aim to schedule and analyse the AMR task set under static priority assignment in global scheduling algorithm in order to cater predictability problem.

ACKNOWLEDGMENT

The authors would like to thank profoundly to the Zamalah Scholarship from UTM, the Research Grant University (RUG), ScienceFund and Ministry of Science, Technology and Innovation Malaysia (MOSTI) Vote No. 4S064, the Universiti Teknologi Malaysia (UTM) for their financial support and not forgotten, the Software Engineering Research Group (SERG) and EReTSEL lab members for their help.

REFERENCES

[1] M. H. Klein, "A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time System", Boston: Kluwer Academic Publisher, 1993.

[2] C.L. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment", Journal of the ACM, vol. 20 (1), pp. 46-61, 1993.

[3] T. P. Baker and S. Baruah, "Schedulability analysis of multiprocessor sporadic task systems, Handbook of Real-Time and Embedded Systems, Chapman Hall/CRC Press, 2007.

[4] T. Wu and S. Jin, "Weakly hard real-time scheduling algorithm for multimedia embedded system on multiprocessor platform", 1st IEEE International Conference on Ubi-Media Computing, Lanzhou, pp. 320-325, August 2008.

[5] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m,k)-firm deadlines", *IEEE Transactions on Computers*, vol. 44(12), pp. 1443-1451, December 1995.

[6] Y. Kong and H. Cho, "Guaranteed scheduling for (*m,k*)-firm deadlines-constrained real-time tasks on multiprocessors", 12th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 18-23, 2011.

[7] G. Quan and X. Hu, "Enhanced fixed-priority scheduling with (*m.k*)-firm guarantee", Proc. of 21st IEEE Real-Time Systems Symposium, Orlando, Florida, USA, pp.79-88, 2000.

[8] Z. Ming and Z. Hai, "A new fixed-priority scheduling algorithm with (*m,k*)-firm guarantee", International Conference on Electronic Computer Technology, pp. 92-95, 2009.

[9] M. Ould-Sass, M. Chetto and A. Queudet, "BGW: A novel QoS model for firm real-time computing systems", Proceedings of the 11th ACM International Symposium on Mobility Management and Wireless Access, New York, USA, pp. 93-100, 2013.

[10] S. Davari and S. K Dhall, "On a real-time task allocation problem', In Proceeding of the 19th Hawaii International Conference on System Science, Honolulu, January 1985.

[11] B. Anderson, "Static-priority scheduling on multiprocessors", PhD Thesis, Department of Computer Engineering, Chalmers University of Technology, Göteberg, Sweden, 2003.

[12] S. Lauzac, R. Melhem and D. Mosse, "An efficient RMS admission control and its application to multiprocessor scheduling, In Proc. Of the IEEE Int'l Parallel Processing Symposium, Orlando, Florida, pp. 511-518, March 1998.

[13] G. Bernat and A. Burns, "Weakly hard real-time systems", IEEE Transactions on Computers, vol. 50(4), pp. 308-321, April 2001.

[14] D. N. A. Jawawi, S. Deris and R. Mamat, "Enhancement of PECOS embedded real-time component model for autonomous mobile robot application," IEEE International Conference on Computer Systems and Applications, pp. 882-889, 2006.