# Scheduling Method with Considering RTOS's Overhead Property

Kazuki Hasegawa (s1310056)

School of Information Science,
Japan Advanced Institute of Science and Technology

08/06/2015

**Keywords:** RTOS, Scheduling, Rate Monotonic, Overhead, Context Switch, Slack, Slack Reclaiming.

## 1 Introduction

In real-time embedded system development, the system is increasing complicated. For supporting complexity system development, Real-time Operating System(RTOS) is used, and helps build an efficient system. By using RTOS, developers do not need to build a function for each of real-time or multitask mechanisms such as performing context switching. However, if the developer uses RTOS functions, system will generate overheads. These overheads are not directly related to the application algorithms.

An aim of this study is propose a new scheduling method considering RTOS's overhead property. My study is focused on RTOS processing for switching tasks. The processing is divided into two types. One is based on activation of tasks. The other is based on terminating task and switching. This research suggests a new scheduling way including task switch. Furthermore, the paper proposes how to improve response time of aperiodic requests. It can achieve that a system uses capacities for overestimated overheads for aperiodic tasks.

## 2 Related Work

According to the reference book[1], many researchers have proposed real-time scheduling algorithms, representative ones are background scheduling, Polling Server (PS), Deferrable Server (DS) and others. In this study, an environment is based on Rate Monotonic scheduling. In addition, this paper deals with both periodic processing and aperiodic requests. They involve task switching.

Polling Server is used for management of aperiodic requests, but this server can only execute at periodic time. Therefore, Polling Server is fitting for management of periodic request. Deferrable Server is also used for management of aperiodic request. Unlike Polling Server, this server can execute any time when the server's conditions are agreeable.

However, these existing scheduling algorithms have a weak point. That is, these algorithms do not include their own processing. An actual system needs to terminate task execution, decide which task to run next, and switch to the new task. However, the existing ones do not include the processing for them. Therefore, the condition for schedulability cannot be derived when the system runs in an actual environment. This paper proposes fills the difference between actual and ideal environments.

## 3 Proposed Method

This paper introduces new units of time: system tick and processing tick. System ticks are used for defining tasks' release times and deadlines. Processing ticks define actual execution times.

Using both the ticks, our study focuses on how to manage activation of periodic tasks which is periodic. As described in the previous section, Polling Server can be used for periodic management, so our solution uses this server for periodic requests. This management server for activating tasks is named Periodic Management Server(PMS). On the other hand, management of finishing and switching tasks is an aperiodic event. Therefore, our solution uses Deferrable Server. In a system, Deferrable Server is equired to have the highest priority, since tthe system blocks if the server cannot execute the management of the occurring aperiodic events. This management server is named Aperiodic Management Server(AMS). Both the servers are based on RM, thus using these servers makes it possible to discuss the schedulability condition in the RM field and developers can derive assurance about scheduability condition.

Unfortunately, both the servers need to have more capacity than the minimum amount necessary to manage the target periodic tasks The system generates a waste of time. This chapter proposes how to calculate the amount of time. This technique is Slack Reclaiming. This slack is reused for aperiodic requests. Using this solution, this study aims to improve response times of aperiodic tasks.

## 4 Evaluation of Proposed Method

This study uses a periodic task set generator and an aperiodic task set generator for evaluating. When creating random information, both the generators use the same randomiser for exponential distribution. the periodic randomiser requires random seed, average of period, average of execute time, the aperiodic randomiser requires average of interval time and average of execution time. By using this randomiser, the periodic task set generator creates task's period, execution time and average of execution time, the aperiodic task set generator creates arrival time and execution time.

Simulator is used for evaluating responsiveness of aperiodic requests. The simulator is required to input periodic task set information, aperiodic request list and simulator environment. After inputting, it outputs information of schedule: can or cannot protect deadline, average amount of aperiodic responsiveness time and load of simulation.

As a result of simulation, aperiodic responsiveness is improved when aperiodic demand rate is lower, while it does not depend on the system load. When the generator change aperiodic load rate, it change only execute time without controlling arrival times. Therefore, when aperiodic load rate is very lower, this is more efficient, because it increases the proportion of slack processing time in aperiodic request.

Furthermore,aperiodic responsiveness is also improved when the system runs with relatively high RTOS overheads. The average of improvement is 1.49 % when the overhead amount is 1 processing tick. When the overhead amount is 2 processing ticks, the average is 2.79 %. Finally when the overhead amount is 4 processing ticks, the average reaches 4.98 %.

## 5 Conclusion and Future Work

This paper proposes a new scheduling technique with including task switch as a RTOS function, and how to improve response times of aperiodic request by using overestimated server capacity. By using these methods, aperiodic response times are reduced by up to 31.56 % compared to a non-server management environment.

For simplification, this study involves only periodic tasks in schedulability analysis. As future fork, in order to approach the real system, this schedule method needs to include grantee structure for aperiodic requests.

## 参考文献

[1] G.C.Buttazz, ”Hard Real-Time Computing Systems: Predicatable Scheduling Algorithms and Applications”, 3rd edition, Springer, 2011.

[2] C.L.Liu, J.W.Layland, “ Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment ”, Journal of the Association for Computing Machinery, Vol. 20, No. 1, pp. 46–61, January 1973.

[3] J.P.Lehoczky, L.Sha, J.K.Strosnider, “ Enhanced Aperiodic Responsiveness in Hard Real-Time Environments ”, Proc. of IEEE Real-Time Systems Symposium, pp. 261–270, December, 1987.